WebCOM: A Communication Framework for CSCW Applications

MD ASRI NGADI

Doctor of Philosophy Degree

Supervisor: Bernard Doherty

ASTON UNIVERSITY

April 2005

To my beloved father and mother:

Ngadi Tahir

Salamah Mohd Narawi

To my beloved wife and kids:

Zairunnizam Ehwan

Muhammad Asyraf Afiq

Muhammad Asyraf Ammar

Muhammad Muzzammil

Fatimah Nur

Acknowledgements

I would like to thank my supervisor, B.S. Doherty for his invaluable advice and assistance throughout the course of this research. Also to all staff of Computer Science Department, Aston University, many thanks for their support. I would also wish to acknowledge the University Technology Malaysia and Public Service Department of Malaysia for providing the financial support for my study.

Aston University

WebCOM: A Communication Framework for CSCW Applications

Md Asri Ngadi

Doctor of Philosophy Degree, 2005

Thesis Summary

CSCW applications are used by people working in a group to accomplish the group goal. However, many CSCW applications have failed to provide sufficient support to users because of limitations in the communication framework, which cause limited accessibility, portability, reliability, performance, and scalability. This research has investigated a solution to some CSCW problems by enhancing Web multicasting to create a new communication framework for CSCW applications. This framework is called WebCOM. WebCOM was implemented using the Java language and Java APIs. The framework uses a Web browser as user interface to enable users in a group to access a shared-application from anywhere and at anytime. A Web browser on the user side can serve as both sender and receiver. The main transmission method used to deliver data between users in a group is multicasting. The framework has been tested by implementing a CSCW prototype on top of the framework to confirm the function of the framework in a CSCW context. The framework also has been tested for performance by measuring response time, throughput, and reliability. WebCOM showed improved performance when compared to the same prototype CSCW system running on a client/server model using unicasting. The architecture of the framework allows tentative extension of conclusions of this work to a wider range of distributed applications.

Keywords: Computer Supported Cooperative Work, World Wide Web, IP multicast, Web-multicasting, workgroups application

Contents

List of Figures

List of Tables

List of Abbreviations

ACK	-	Positive Acknowledgement
API	-	Application Programming Interface
CSCW	-	Computer Supported Cooperative Work
DNS	-	Domain Name Service
HTML	-	HyperText Markup Language
IP	-	Internet Protocol
LAN	-	Local Area Network
LBRM	-	Log-based Receiver Reliable Multicast Protocol
LRMP	-	Light-weight Reliable Multicast Protocol
MBone	-	Multicast Backbone
NAK	-	Negative Acknowledgement
RTP	-	Real-Time Transport Protocol
RTMW	-	Real-Time Multimedia Web
RMTP	-	Reliable Multicast Transport Protocol
SRM	-	Scalable Reliable Multicast
TRAM	-	Tree-based Reliable Multicast
TMTP	-	Tree-based Multicast Transport Protocol
TCP	-	Transmission Control Protocol
UDP	-	User Datagram Protocol

LIST OF ABBREVIATIONS

URL	-	Uniform Resource Locator
WAN	-	Wide Area Network
WBRM	-	Web-Based Reliable Multicast
WYSIWIS	-	What You See Is What I See
WWW	-	World Wide Web

List of Terms

Bandwidth	-	The amount of data that can be transmitted in a fixed amount of time. The bandwidth is usually expressed in bits per second (bps) or bytes per second (Bps).
Bottleneck	-	A bottleneck refers to congestion that causes delay in transmission of data over a network. Bottlenecks affect network performance by slowing down the flow of information transmitted across networks.
IP	-	Internet Protocol(IP) specifies the format of packets, also called datagrams, and the addressing scheme.
Latency	-	The amount of time it takes a packet to travel from source to destination. Together, latency and bandwidth define the speed and capacity of a network.
Packet	-	A piece of a message transmitted over a packet-switching network. One of the key features of a packet is that it contains the destination address in addition to the data. In IP networks, packets are often called datagram.

LIST OF TERMS

Throughput	- Throughput is the amount of data moved successfully from one place to another in a given time period. Typically, throughputs are measured in kbps/kBps (kilo bits per seconds / kilo bytes per seconds), Mbps (mega bits per seconds) and Gbps (giga bits per seconds).

Chapter 1

Introduction

This thesis describes an extended communication framework, *WebCOM*, which addresses issues of performance, portability, scalability, accessibility and reliability in groupware implementation over the Internet. The objectives of WebCOM are to provide quick response for group communication on the Internet, portable to any platform, scalable to support a large number of users, with continued operation of the system despite some participants or servers having failed, and to provide reliable data transmission over the network.

WebCOM has been built using the Java language and Java APIs which run on a hybrid architecture. A prototype CSCW application has been developed and implemented on top of WebCOM and on top of a "normal system"[1] to test the above objectives. Experiments have been done for both systems in the same testbed. The systems were evaluated by measuring performance and QoS[2] metrics; the metrics used were response time, throughput, and packet loss.

1.1 Motivation

In conventional working, people work individually, attend frequent meetings to make a decision on certain issues, and spend much time, energy and money on travel and

[1] client/server unicasting system
[2] Quality of Service

physical meetings. Nowadays, people have moved towards a group style of work to accomplish their task (Encarnação *et al.*, 1998). The field that studies the use of computer technologies to enable people to work together by cooperating directly using shared resources in a small, closed group, is called CSCW (Computer-Supported Co-operative Work) (Leung and Cheung, 1999; Coulouris *et al.*, 2001). The benefits from working in a group are:

- people spend less time attending meetings, and more time completing the task,
- a decision can be distilled from many ideas in a group,
- a big task can be accomplished more easily by sharing the work among people in a group.

From these benefits, there has been demand for groupware applications to be developed to support daily tasks in a group. The efficiency of completing a task in a group is normally enhanced by tools such as a *drawing* tool, a *construct message* tool, and *multimedia conferencing* tools (Wiil *et al.*, 2003). Other tools may be used depending on the user requirements. The efficiency of groupware applications is measured by two different aspects:

- groupware communication:
  - faster response time (Lin *et al.*, 2003),
  - reduced network transmission load (Lin *et al.*, 2003),
  - low error rates (Borella *et al.*, 1998), and
  - low delay for data transmission (Ingvaldsen *et al.*, 2000),
  - shorter access time (Hermanns and Engbrocks, 1994)
- human-machine interaction measures, such as ease of use (Agostini *et al.*, 2002)

Until now, a number of CSCW applications have been used and are being developed to support tasks for different groups of people, for example text conversation using a chat system, and a shared editing environment using a shared whiteboard system. Although many applications have been used and developed, it has been reported that a number of applications have failed to serve the needs of users for a variety of reasons (Bullen and Bennett, 1990; Grudin, 1989; Lynne Markus and Connolly, 1990; Orlikowski, 1992; Hummel *et al.*, 1996). As a result, applications were abandoned by users, who returned to conventional ways. In the following section, a number of issues that caused some CSCW applications to fail to serve the needs of user are described.

1.2 The Problems

Many CSCW applications have been developed and are being used, but a number of issues still remain open for better solution. Most of the issues discussed concern providing flexible tools to the users, where *flexible* means that users can access the tools from anywhere and anytime within a group, with good response and reliability. The main problems discussed in this thesis concern the following efficiency issues discussed earlier (Section 1.1):

- **Performance**

  Most CSCW applications have faced difficulties in supporting a high number of users and high bandwidth, which leads to performance problems (Ingvaldsen *et al.*, 2000; Hallin *et al.*, 1999; Parr and Curran, 1999; Keshav and Paul, 1998; Matta and Eltoweissy, 1998). CSCW applications are traditionally implemented using unicast protocols which function well for two or three users; however, performance degrades quickly, as the required bandwidth increases when the number of user increases (Zabele *et al.*, 1994). This has led to research into multicasting, which is seen as more suited to one-to-many communication.

- **Reliability**

  The need for reliability arises from the fact that the communication service needs to function in environments such as the Internet where data packets may be delayed or lost, and destinations may become temporarily unreachable in the system. In collaborative systems, processes need to reliably transmit messages from a source to multiple destinations. This is significantly more difficult for multicasting compared to unicasting, which has been successful over the years (Hall *et al.*, 1996). Although multicasting offers good performance for group work environment, it faces the issue of reliability.

However, many works have sought to provide reliable multicasting and it is under active research. Some of these works are reported here, namely Levine (1996), Miller (1997), Wright *et al.* (2000), Basile *et al.* (2003) and Takenaka and Sasaki (2004). These works discuss providing reliability to multicasting.

- **Portability, Availability and Scalability**

  Some CSCW applications are developed for particular computing platforms and thus are usable only within a specific organisation supporting those particular platforms (Broll *et al.*, 2000; Lee *et al.*, 2000; Chaun, 1999; Bentley *et al.*, 1997; Dix, 1996). Limiting the ability of groupware applications to work on any platform sometimes will reduce the efficiency of the work (Wilde, 1994). However, many applications have moved towards a Web environment style which provides easy access and is able to run on any platform and environment that support Web technologies. On the other hand, when a large number of users are accessing a single central point of a server, a bottleneck will occur. This will cause the server to be unable to cope with requests from clients and it will fail. Thus a client/server has limited scalability[3] and availability.

  Most CSCW applications do not scale well to support changes in the number of users (Matta and Eltoweissy, 1998; Zabele *et al.*, 1994). Certain data streams in the collaboration system may need to be delivered to potentially large numbers of users, and the communication service should allow for this in a cost effective manner, i.e., communication protocols for certain data streams must be scalable. With good scalability the data delivery is cost-effective even when there

[3]scalability definition by Coulouris *et al.* (2001)

are a very large number (hundreds, thousands, or even tens of thousands) of destinations that the data needs to be delivered to. The cost metric used in this thesis to determine the scalability of WebCOM is primarily number of messages.

1.3 The Solution

The solution proposed in this thesis has combined aspects of previous work in the area to produce a solution that aims for:

- **efficiency of performance**

  It is necessary to ensure all communication and transaction activities satisfy the users' needs especially in terms of performance speed. The performance is affected by many factors, for example network architecture, application design, network traffic, and many more. However, this thesis concentrates on providing a communication framework that *can reduce* network traffic by implementing a particular network architecture and application design. This thesis has used a hybrid architecture, using multicasting as a main communication method for efficiency of performance in group communication.

  There are many different architectures available in distributed systems such as central, replicated, distributed, peer-to-peer, and hybrid (Zabele *et al.*, 1994; Balter *et al.*, 1995). These architectures have their own advantages and disadvantages, which are discussed in Appendix B. The main advantage of using hybrid architecture is its design flexibility. For example, a hybrid architecture of client/server and peer-to-peer provides easy management and gives fewer bottlenecks.

  According to Maihöfer (2000), multicasting provides good scalability for a large number of participants in a group. If the unicasting method is used, every participant is required to have their own connection for every transmission and separate copies of the same data are required for every participant. Multicasting only requires one copy to be sent, which is locally distributed to every participant.

There are three ways to implement multicasting on the Internet: ATM, IP multicast, and Ethernet. Among these, this research has chosen Ethernet to implement multicast in the Internet because:

- it is easy to implement
- it is widely available
- no special equipment is required to implement multicasting, such as special routers or network interface cards.

However by choosing the Ethernet technology, a number of limitations occurs in the system such as limitation on number of users in a groups supported, packet size supported, and type of stream supported.

- **reliability of data transmission**

  Reliability of transmission can be achieved by implementing mechanisms in the transmission processes that check incoming and outgoing data streams. This study has used a reliable multicast protocol to ensure reliability of data being sent using the multicast method. A few studies have implemented a reliable multicast protocol in their application: Pavilion (McKinley *et al.*, 1999b) uses WBRM (Web-based Reliable Multicast) and the MASH project(McCanne *et al.*, 1997) uses SRM (Scalable Reliable Multicast). This study has adapted the LRMP (Lightweight Reliable Multicast Protocol) used by Liao (1997) to distribute Web documents over the Internet. This issue is discussed in Chapter 2, Section 2.5.2 in more detail.

- **efficiency of support for large number of users**

  It is necessary to ensure all interested participants are able to participate in a group discussion or meeting. This thesis has used multicasting to provide good scalability for a large number of users. The technology chosen for the system influence the capability of overall system to support a large number of users effectively in term of performance. However, the number of users that chosen for this research for testing is under the scope of CSCW applications.

- **ease of use in term of accessibility and portability**
  It is a good practice to allow users to access and use a CSCW application from any location and platform. This thesis has used the World Wide Web (WWW) and the Java language to achieve these capabilities. The World Wide Web (Berners-Lee *et al.*, 1994) offers a globally accessible platform independent infrastructure. Although WWW technology has a number of limitations, imposed by the architecture of the Web, there are also definite advantages such as easy cross platform use, good integration with the typical work environment of users, and easy integration with other technologies such as Java and CGI.

In conclusion, the study reported in this thesis has combined WWW technology and multicast technology by proposing WebCOM as an alternative communication framework to implement distributed applications in the Internet, with the work in this thesis offering a solution to the problems of CSCW mentioned in Section 1.2. A prototype of a CSCW application has been developed to test the ability and efficiency of WebCOM. The ability is tested by implementing the prototype on major platforms such as Windows, Unix, Linux and Macintosh, and implementing it within a LAN and WAN. The efficiency is tested by measuring the performance of a generic CSCW application under WebCOM.

1.4 The solution in the context of CSCW

The solution proposed is to address the CSCW issues that have been discussed in previous section. The solution proposed is only in the context of CSCW. Primarily, the solution proposed is focused on improving communication in CSCW applications in term of efficiency, reliability and scalability. Other issues such as security, heterogeneity, and consistency are not addressed in this thesis.

This study has sought to improve efficiency, reliability and scalability in CSCW with a number of approaches:

- to improve the accessibility and portability, the system has implemented using WWW technology.
- to improve the reliability, the system has adapted the Light-weight Reliable Multicast Protocol (LRMP) into the communication design.
- to improve the scalability and efficiency, the system has implemented using multicasting.

The system has integrated the above approaches to produce one system that provides efficiency, reliability and scalability in the CSCW communication. In order to measure the objectives, a generic prototype of CSCW applications has been developed and implemented on the proposed system. The performance of the proposed system has been compared with performance of a prototype of the same CSCW applications implemented in a "normal system" which uses unicasting communication and client/server architecture. The results from both systems are analysed to show that the proposed system offers clear improvement in the chosen performance measures. The proposed system is named WebCOM.

1.5 Aims of thesis

The work described in this thesis can be categorised as problem-solving research. The problems addressed are performance, scalability, accessibility, portability, and reliability of groupware applications. In order to address these problems, the work in this thesis proposes WebCOM, which extends existing communication frameworks by:

1. merging the client/server and peer-to-peer model to produce the hybrid architecture for WebCOM.
2. integrating Reliable Multicast Protocol (RMP) into the implementation of WebCOM to provide reliability of data transmission using multicasting over network.

3. supporting groupware application tasks over the Internet to give quick response and be reliable and scalable.

4. extending the scalability of existing technology to manage a large number of users in a group.

In addition, this research will:

1. develop a generic CSCW application that runs on the WebCOM framework. Then, a test will be done on the CSCW application to confirm it works under WebCOM.

2. test the performance of WebCOM in a collaborative environment. A comparison will be done by running the same CSCW application on a client/server model using unicasting.

1.6 Scope of Thesis

WebCOM has been developed to address issues of performance, portability, scalability, accessibility and reliability in groupware implementation over the Internet. However, the research has some limitations:

- WebCOM deals with small simultaneous users in a group and does not deal with high number of simultaneous users in one group.
- WebCOM tests only on basic text data and does not tested on streamed data.
- WebCOM deals with light-weight reliability and does not deal with high reliability data transfer.
- WebCOM deals with generic functions of CSCW applications and does not deal with complex functions of CSCW applications.

1.7 Research Methodology

Research methodology for this study involved a number of activities: literature review, analysis and investigation, prototyping; experiment, measurement and analysis.

1.7.1 Literature Review

The literature review covers three main areas:

- **CSCW**

  Problems in the CSCW area were investigated to identify the research area. The problems include issues of implementation and design of CSCW application over the Internet. From these issues, available techniques and methods were investigated to address the identified issues.

- **World Wide Web**

  World Wide Web (WWW) technology was investigated for its ability, suitability, and preferability in support of groupware applications over the Internet. Investigation of integration between WWW and multicasting was also done.

- **Multicasting**

  The ability of multicasting to integrate with World Wide Web technology, hybrid architectures, and platforms was investigated.

1.7.2 Investigation and analysis

A number of problems were identified in the CSCW research area. The important and interrelated problems in CSCW were combined. This study has selected one of the combined problems as a main issue to be discussed in this thesis.

The literature investigation was undertaken to determine possible approaches to solve the problems identified above in CSCW. The approaches were analysed by making a comparative study to solve the problems. Then, a suite of approaches that might solve the problems was chosen.

1.7.3 Prototyping

The chosen problems and approaches for this study were tested by implementing them in a testbed. A prototype for the real implementation of groupware applications has been developed and a testbed has been set-up. Generally, the prototyping for this research involved two tasks:

- **Prototype development**

  A generic CSCW application prototype was developed using the Java language. These developments involved the Java libraries and packages. The prototype emphasis is on the communication and interaction of groupware application over the Internet, and *not* the design of interface or applications.

  Two prototypes were developed in this research. The first prototype was a generic CSCW application that runs on WebCOM. The second prototype was a generic CSCW application that runs on a client/server unicasting model, which is treated as a "normal system" in this thesis. The reason behind this was to evaluate WebCOM compared to the "normal system" using client/server unicasting on the same testbed, which gives a fair comparison between two systems.

- **Testbed Setup**

  A network environment was set-up as a testbed to run the prototype. It used the established Aston University network that included servers and clients. The test required many machines as participants of groupware application to create a collaborative environment. The testbed was used to test the prototype that runs on WebCOM and on a "normal system".

1.7.4 Experimental, measurement and analysis

Tests have been done to decide how well the proposed communication framework, WebCOM, meets the objectives of this research set out in Section 1.2: performance, reliability, scalability, portability, and availability. In order to evaluate the proposed communication framework, WebCOM was tested on three different metrics of performance and QoS (response time, throughput and packet loss). The same method of testing was implemented on the "normal system" to get fair comparison of results for both systems.

Experimental activities were carried out to assess the efficiency of WebCOM for CSCW applications. The efficiency of WebCOM was measured by evaluating the performance and QoS metrics (response time, throughput, and packet loss) for WebCOM on the testbed. In addition, the efficiency of WebCOM was compared with a "normal system" tested on the same testbed. The same performance metric was applied for evaluating the "normal system". From the experimental results, an analysis was made and conclusions drawn.

Three main issues that relate to the experiment on the testbed were considered. These issues are:

- **Accuracy**

  The accuracy of measurement for the testing was considered as an important issue in this thesis because numerical comparisons are to be made. Methods of providing accuracy for the testing were investigated.

- **Clock synchronisation**

  Clock synchronisation is a crucial part of the testing because it involved many machines which have different system clocks. This study investigated possible approaches that might help the problem of clock synchronisation among the machines involved in the testing.

- **Network traffic status**

  The testbed was set up on the established Aston University network which has open connection to the Internet. The best condition for testing is at the time when network traffic is low, so there is less activity on the network.

In addition to the above testing, WebCOM was evaluated on its own ability to support groupware applications on the Internet. For this reason, the prototype was implemented on top of WebCOM in different major platforms such as Windows, Unix, Linux, and Macintosh and different network scale such as LAN and WAN.

1.8 Operational Framework

The operational framework for this research is shown in the Figure 1.1 as follows:

1. The communication framework, WebCOM and a prototype were designed. For each design, a number of issues were considered, for example components to be included, architecture to be implemented, languages to be used for developing the system, and the operating system to be used.
2. A testbed was set-up to implement the system. The testbed consists of network connection, server, clients, and intermediate machines such as switches. The testbed was used to test the system.
3. WebCOM and a prototype were developed using the Java language.
4. A full system that consists of CSCW application prototype and communication framework were implemented on the testbed.
5. Testing was done to make sure the system worked on the testbed. If something did not work, it was re-built.
6. The full system was tested and measured according to the experimental plan.
7. The results from the experiments were analysed and an evaluation made.
8. Finally, a thesis describing the study was produced.

1.9 Research Plan

The research plan for this study can be divided into three parts:

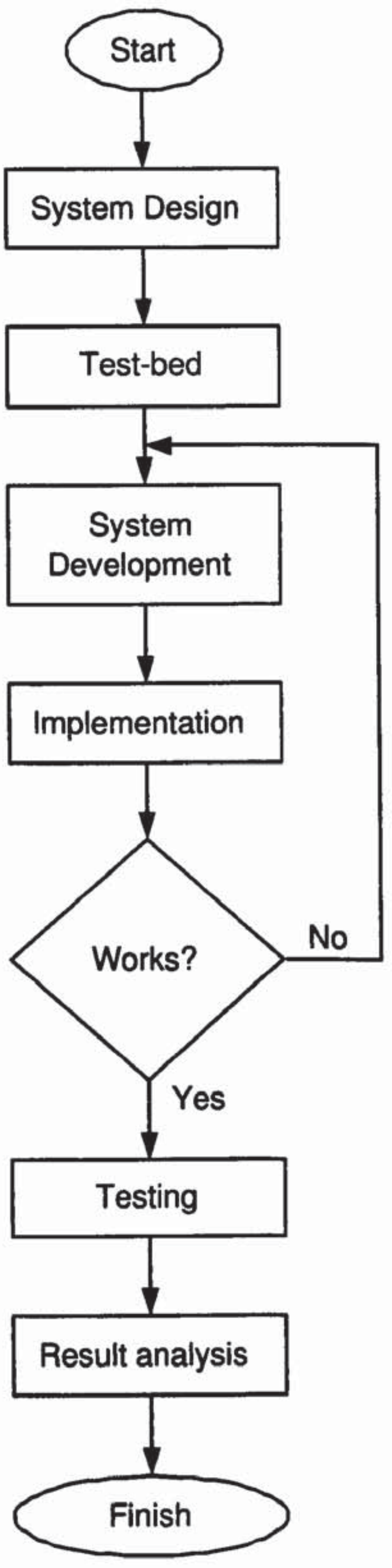


Figure 1.1: Operational Framework of The Research

1. proposal preparation.
2. development and testing phases.
3. writing process.

The detail of research planning can be found in Appendix A.

1.10 Thesis Contribution

The novel aspects of this thesis are:

1. use of a hybrid model to implement WebCOM.
2. use of multicasting as the main network communication in WebCOM.
3. implementation of synchronous and asynchronous CSCW applications on WebCOM.
4. integration of a reliable multicast protocol, Light-weight Reliable Multicast Protocol (LRMP) (Liao, 1997) to the new enhanced communication framework, WebCOM.
5. measurement of the efficiency of WebCOM to support groupware applications in the Internet through performance evaluation.
6. testing of the ability of WebCOM to support groupware applications working on different platforms.

1.11 Outline of thesis

The thesis is organised as follows. Chapter 2 presents a review of the relevant background literature. This involves description of the CSCW requirements issues, previous approaches, and related technology. Chapter 3 is a description and analysis of the proposed method, where the WebCOM framework and design objectives are also set. Chapter 4 describes the prototype developed to perform the ability and efficiency tests. Chapter 5 summarises the experimental results, followed by the interpretation of the results. Chapter 6 evaluates all the work has been done from the beginning of the research until the end. Comments and proposals for further research are also presented in Chapter 6. Finally the conclusion is presented in Chapter 7.

Chapter 2

Literature Review

From Peter Senge, Director of the Center for Organizational Learning at MIT's Sloan School of Management. "...teams, not individuals, are the fundamental learning unit in modern organizations ...unless teams can learn, the organization cannot learn." (Peña Mora *et al.*, 2000)

2.1 Introduction

This chapter starts with an overview of issues in CSCW, looking at the origins and future direction of CSCW. Issues for providing efficient methods to implement CSCW applications on the Internet are discussed. Different approaches have been identified to improve the efficiency of implementing CSCW in the Internet. From the literature, WWW technology and multicast technology give advantages in implementing CSCW applications in the Internet.

2.2 Computer Supported Cooperative Work

2.2.1 Introduction

Computer Supported Cooperative Work (CSCW) or groupware systems allow physically dispersed teams to engage in a common task by providing an interface to a shared workspace(Ellis *et al.*, 1991). There are many definitions available for CSCW. According to Wilson (1991), Computer-Supported Cooperative Work(CSCW) is defined as a generic term which combines the understanding of the way people work in groups with the enabling technologies of computer networking and associated services, techniques, hardware, and software. Hill and Brinck (1994) defined "CSCW is the study of how people work together using computer technology. Typical topics include use of email, hypertext that includes awareness of the activities of other users, video conferencing, chat systems, and real-time shared applications, such as collaborative writing or drawing."

The purpose of CSCW is to provide computer support that facilitates cooperation between users through its' applications and tools (Foley and Jacob, 1995). Many CSCW applications have been developed to support users' daily tasks such as mail, chat system, video conferencing, shared whiteboard and many more. More recently, commercial products such as Microsoft NetMeeting have emerged to support some co-operative editing processes (see `http://www.microsoft.com/windows/NetMeeting/Features/`). To understand the varieties of CSCW applications, many authors have categorised these applications into classes according to certain criteria described in the following section.

2.2.2 Classification of CSCW

According to Condon (1993), many criteria have been proposed to classify CSCW application. A widely adopted taxonomy of CSCW applications is that based on the temporal nature of interactions among participants (Clarence and Wainer, 1994). The interactions can be achieved in a synchronous or an asynchronous way (Lauwers *et al.*, 1990). Desanctis and Gallupe (1987) have illustrated this taxonomy as division of same and different time and place, which requires synchronous versus asynchronous tools. Ngwenyama and Lyytinen (1997) describe another taxonomy in which tools are instrumental, communicative, discursive or strategic. Spellman *et al.* (1997) have categorised CSCW application into two categories: session-centric tools, such as most desktop video conferencing; and document-centric tools, such as Lotus Notes and doc-

Figure 2.1: Categorised Groupware Applications According Place And Time (Grudin, 1994)

ument management systems. These categories have been expanded by Grudin (1994) as shown in Figure 2.1 to acknowledge the fact that both place and time can be the same, different and predictable, or different but unpredictable.

Figure 2.2: Classification Space For CSCW Systems (adapted from Rodden (1991))

Rodden (1991) describes the CSCW classification by space where asynchronous and synchronous interaction can be done with different locations either co-located or remote as shown in Figure 2.2. As shown in Figure 2.3, Butler and Coleman (2003) has classified the collaboration work with group size and the level of the interaction in each group.

Figure 2.3: Model of Collaboration (Butler and Coleman, 2003)

In summary, there are many classifications of CSCW applications and interactions. These taxonomies help to explore the purpose and possibilities of CSCW application tools, the circumstances under which they are useful, and help the users to more easily understand the environment and the requirements of CSCW applications. From these classification, four important elements have been identified: applications type, place and location, the level and type of interaction, and size of participants. The type of application will depend on:

- the place or location, where the application will be used either in a small building, within a LAN, or within a WAN.
- how frequent the interaction and how the interaction will take place, either synchronous or asynchronous.
- how many participants, whether 5 person, 20 person, or more.

2.2.3 Issues in CSCW

CSCW applications were developed to fulfil the users' requirements, which are various, depending on the scope, usage, environment, user's knowledge and perception. Generally, users want a system that is simple and easy to use, gives quick responses, is easy to access from anywhere and at anytime, without any constraints (Raatikainen, 2002; Pentagon, 1997; Sun Microsystems, 1998). There are other requirements in CSCW which are not discussed in this thesis, for example security issues (Bouras *et al.*, 1999, 2000; Geyer and Weis, 2000), high speed technologies issues (Nentwig *et al.*, 1996) and human factors (Dix *et al.*, 1998; Kristoffersen and Ljungberg, 1998; Bourges-Waldegg and Scrivener, 1998). The main issues of CSCW discussed in this thesis are (Jasnoch, 2001; Adoud *et al.*, 2001; Cheung and Chanson, 1999; Matta and Eltoweissy, 1998):

- **Scalability**

  Scalability is required for a large group of users, with frequent changes in user numbers.

- **Reliability**

  Each CSCW application has degrees of reliability depending on the users requirement. Some applications need high reliability and others can tolerate loss of data.

- **Performance**

  Many applications need an adequate performance to process or respond to requests in order to satisfy the users' need. Unreasonable performance will make the users unsatisfied and might lead them to abandon that application.

- **Availability**

  The application should always available when a user wants to access and use it.

- **Accessibility**

  Users are able to access the CSCW applications from any location and platform.

- **QoS guarantees**

  The CSCW applications need to give a level of quality to the users to ensure users are satisfied with the given services.

2.2.4 Related Work

This section discusses related work in CSCW implementation in two areas: **World Wide Web(WWW)** and **multicasting**. World Wide Web (Berners-Lee *et al.*, 1994) technology is a well known technology that distributes information on the Internet. The World Wide Web offers a globally accessible platform independent infrastructure. Although WWW technology has a number of limitations, imposed by the architecture of the Web, there are also definite advantages such as easy cross platform use, good integration with the typical work environment of users, and easy integration with other technologies such as Java and CGI. A number of CSCW application found in the literature are implemented in WWW. For examples, GroupWeb (Greenberg and Roseman, 1996) and CoWare (Lee *et al.*, 2000) have used the WWW as a shared workplaces for their applications and AISA (John Riedl *et al.*, 1997) used the WWW for asynchronous software inspection.

The Multicast Backbone (MBone[1]) (Erikson, 1994) provides the infrastructure for efficient multipoint data delivery on the Internet. mMosaic (Dauphin, 1996) was one of the first tools for sharing Web documents over the MBone. mMosaic is based on a modified version of the NCSA Mosaic browser, which processes incoming resources and multicasts them along with formatting instructions. mMosaic can multicast Web documents together with inline images to remote Mosaic users. It uses a repeated transmission scheme by simply re-multicasting data every 4 seconds, but users can cache data segments from previous cycles. At each retransmission, remote users only need repair missing segments. It is currently being developed and is an extended version of the WWW browser, XMosaic. Initial tests show that mMosaic works well with HTML-pages and smaller images, but the distribution delay is too large with

[1]a set of multicast-capable routers on the Internet

bigger images.

MCM(Multicast Mosaic) (Touvet, 1996) is a tool specially for multicast of HTML slides. While initially used with the NCSA Mosaic browser, support for other browsers (e.g. Netscape) was added in a newer version. MCM uses a master-slave model. A multicast daemon (master) packs all slides into a single package and repeatedly multicasts this package in the hope that at least one error-free copy can get to remote sites. When the master loads a Web page, only the URL of this page is multicast. Upon receipt of the URL, those users who have already received the package can visualise the corresponding document. The main problem with MCM is its repeated transmission scheme which is not suitable for real time applications.

Another tool designed to work with Mosaic is Web-cast (Burns, 1995). Webcast was designed especially to work with the X-Mosaic browser through its CCI[2] interface. It is used for sharing Web documents over the MBONE by either multicasting URLs or HTML documents. Multicast can be carried out either on HTML documents or only on URLs. Web-cast was built on the Reliable Multicast Protocol (RMP) to cope with network transmission error, but RMP is now considered to be unsuitable for wide area network based applications due to its token ring control scheme and because only the original sender of the data could repair lost packets (Liao, 1997).

WebCanal (Liao, 1997) is a proxy-based synchronous browser program written in Java. This application is designed for exchange of Web documents, basically of the text/html MIME type, on the Internet, in multicast mode among a group of users, or in unicast mode between two users. Changes in the current URL displayed by the master browser are detected by a proxy, which parses files to identify inline images, and transmits the data to the set of receivers. Since WebCanal operates only as a proxy, local control operations (such as Back and Forwards browser commands, and loading of local files) present difficulties.

mWeb (Parnes *et al.*, 1997) is an ongoing project that uses the Web browser as

[2]Common Client Interface

a multimedia slide presentation medium. mWeb is similar to WebCanal in that it is proxy-based and uses IP multicast to distribute Web resources. mWeb uses SRM/RTP (Scalable Reliable Multicast/ Realtime Transport Protocol) to offer real time reliable services to applications. mWeb is an effort to offer a multicast Web tool through multiple browsers and seems to be a step towards providing reliable multicast tools.

Shin (1997) has extended the World Wide Web architecture to include protocols, addressing schemes, and page formats to merge multicast capability into the Web using the HotJava protocol handler. Previous work has been extended by Shin and Lee (1998) and Shin *et al.* (1998) by integrating other protocols to allow Web users to join a session and receive audio/video seamlessly.

Real-Time Multimedia Web (RTWM) (Shin *et al.*, 1998) is a Web-based multimedia system for MBone, which enables Web users to join an MBone session and receive real-time audio/video without helper applications. RTMW used Web browsers at the receiver side and MBone tools[3] running at the sender side, and used a Web server.

WebClass (McKinley *et al.*, 1999a) is a multicast browser tool written in Java. WebClass has implemented the multicast browser using reliable multicast communication, WBRM (Web-Based Reliable Multicast) running on top of IP Multicast, to distribute control information and Web resources from a "master" browser to multiple "client" browsers.

In conclusion, there are many ways to implement multicast applications in the Internet. However, in the CSCW field, it seems little work has been done to implement multicasting in CSCW applications. Much work on multicasting focuses on real-time transmission, but there is less work done on collaborative environments, especially in synchronous and asynchronous applications such as CSCW applications. That is why, this study has proposed to use the WWW and multicasting to form a communication framework known as WebCOM and run a CSCW application on it and to determine:

- the suitability of WebCOM to run CSCW application

[3]such as sdr, vic and vat

- the ability of WebCOM to support CSCW application
- the efficiency of WebCOM in term of performance.

2.3 From Unicasting to Multicasting

The most used communication method in distributed applications on the Internet are commonly based on a single sender, single receiver model known as *unicasting* (Aggarwal *et al.*, 2000). However, this model is not efficient if large amounts of data or time-sensitive data is transmitted to multiple recipients (Labonté and Srinivas, 2000). Another method that can be implemented in these environments is multicasting. The following section (Section 2.4) describes the comparison between these methods. From discussion in Section 2.4, multicasting provides better support to multiple recipients where data is sent to a specific group and is only duplicated for delivery purposes when necessary.

Multicasting enables a packet sent from one sender to be received by all group members simultaneously. It permits sending packets to a group of receivers without duplicating them at the source. Multicast packets are duplicated automatically by multicast routers that are close to the receivers in the network. With this feature, multicasting saves network bandwidth and offers an efficient communication technique for group communication (Maihöfer, 2000). In order to receive multicast packets, receivers join the multicast group(s) to which multicast packets are delivered. Receivers also can leave the multicast group in order to terminate the delivery of multicast streams to their nodes. The "join" and "leave" operations are performed dynamically at run-time independently of the sender. Now, IP multicasting has became the de facto standard of multi-point group communication in the Internet (Al-Shaer and Tang, 2001).

In the next few sections, a summary of the available technologies of data transmission over network is provided. This is followed by a discussion of multicast technology and its implementation.

2.4 Data transmission over the Internet

Figure 2.4: Three Different Methods of Data Transmission Over the Internet (Dörries and Zier, 2001)

There are three fundamental methods for transmitting data on the Internet, as illustrated in Figure 2.4:

- IP Unicast
- IP Broadcast
- IP Multicast

IP Unicast transmission is designed to transmit data or a packet from a sender to a single receiver, as shown in Figure 2.4(a). IP Broadcast transmission is used to send data from a sender to an entire subnetwork, as illustrated in Figure 2.4(b). IP Multicast transmission is designed to enable the delivery of data from a sender to a set of receivers that have been configured as members of a multicast group in various scattered subnetworks, as shown in Figure 2.4(c) i.e. to broadcast to a prescribed set of users rather than to all on the network.

2.4.1 IP Unicast

Most distributed applications use the unicasting method to transmit data over the Internet. In unicasting, connection-oriented stream transports are used for distributing data to each receiver individually. This means receiver and sender need to establish a connection to an identified server before they can communicate each other. When data transmission is needed, the application will duplicate the data for each receiver that is connected, and transmit each copy of data required by the receiver through the established connection. If the number of users increases, the usage of networks also increases. If there are N receivers, the sender needs to send N copies of data. This will cause network traffic to increase and degrade the quality of applications.

2.4.2 IP Broadcast

A second method of data transfer is to broadcast the data. This entails each bridge or router forwarding the received packets on all interfaces with the exception of the path from which the packet was received. This is similar to a television broadcast, in that the signal will be retransmitted without regard to which receivers are interested in the data. This results in the indiscriminate use of bandwidth, as all destinations will receive the packets, which must traverse all data transmission routes, even if there are no destination addresses downstream.

2.4.3 IP Multicast

IP multicast (Deering, 1989, 1991) is one method that allows distribution of data from one sender to many receivers. A multicast datagram is delivered to all members of its destination host group with the same "best-efforts" reliability as regular unicast IP datagrams. The membership of a host group is dynamic i.e. hosts may "join" and "leave" groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. Allowing any host to "join" or "leave" at any time makes it easy to scale for a large number of participating hosts. The extendibility of the group size makes the IP multicasting model very attractive from the perspective of *scalability* (Hardjono and Cain, 1999).

IP Multicast transmission provides for sending the data from a sender to multiple receivers, but unlike IP Unicast, the number of identical copies that are sent is limited. IP multicast leads to efficient bandwidth utilisation for one-to-many and many-to-many communication. IP multicast also reduces the load on the sender because a transmission is performed once per group. Senders simply send their packets to an abstract *group address* and receivers express their interest in receiving these packets by joining the corresponding multicast group address through a group membership protocol (Fenner, 1997).

The abstract group address is selected from the special class D range (between 224.0.0.0 and 239.255.255.255) of IPv4 addresses, and serves as a handle or key to the entire multicast group, thus obviating the need for higher-level applications to maintain explicit membership lists. All receivers are configured as members of the same multicast group. The sender sends an IP packet to a multicast address, and lets the network forward a copy of the packet to each group of hosts. Multicast is not connection-oriented; the sender sends data to multiple receivers over UDP (User Data Protocol). The UDP, unlike TCP, makes only a *best effort* to deliver data. If a transmission error occurs, the packet is discarded.

In multicasting, each user needs to join a specific address (IP address class D) as a shared address for a group. Once established, if any user wants to submit data, that user need only send one copy to that shared address. All the transmission processes will be handled by a nearest router from a sender to a nearest router to receiver(s). Intermediate router(s) will make a copy of the data and pass to the neighboring router. The nearest router to the receivers will make copies of that data to the number of the receiver on a subnet. If there are N receivers in a group, a sender only need send one copy of data.

2.5 Implementation of Multicasting

Three technologies currently dominate the high-speed networking market (Dörries and Zier, 2001):

- Packet-Over-Sonet (IP Multicast)
- Asynchronous Transfer Mode (ATM)
- Ethernet (Gigabit Ethernet)

2.5.1 Packet-Over-Sonet (IP Multicast)

IP multicast (Deering, 1991) uses special IP addresses (Class D) for data distribution. Multicast receivers register with the Internet Group Multicast Protocol (IGMP)[4] at their nearest IP multicast router. The router uses special IP multicast routing protocols (like DVMRP and PIM) to dynamically build up *distribution trees*[5] for multicast groups. The distribution trees are mostly sender-based, but some protocols also use core-based distribution, where a sender sends the packets first to a core (which may be a single IP router), from which a distribution tree delivers the packets to all receivers. This is a good solution for sparse groups with many senders.

Modern IP routers[6] support multicast in hardware and are able to forward multicast packets at or near line rate, so it is no problem to build a high-speed IP multicast network from scratch. But in many existing networks, older routers and slower links may exist and the central problem is to restrict the multicast distribution and prevent a flooding of the multicast packets in the whole multicast network. The distribution

[4]Appendix C.5.1

[5]Appendix D.1.2

[6]most modern routers have a function to enable multicast service

of the multicast packets may be limited over the *time to live* (TTL)[7] field in the IP header, but a more secure method is to use special multicast groups and access lists if the IP routers support this feature.

The idea of IP Multicast was first implemented in the multicast backbone (MBone). The Mbone began as a small virtual network on top of the existing Internet. Workstations served as routers, and multicast packets were exchanged by encapsulating them in unicast packets, termed *tunneling* (Aggarwal *et al.*, 2000). The MBone today covers thousands of networks with growing portions of the MBone consisting of entire administrative domains with production routers handling both unicast and multicast routing.

2.5.2 ATM

The multicast distribution in ATM networks is based on point-to-multipoint connections. The multicast distribution tree realises the shortest paths between the sender and each receiver. Only the sender may add new receivers. While several extensions to this basic concept exist such as receiver-initiated point-to-multipoint and multipoint-to-point connections, they are not available in most networks. ATM point-to-multipoint connections are supported in hardware in almost all modern ATM switches, and ATM point-to-multipoint traffic is limited only by available trunk bandwidth and end-system limitations.

Multicast distribution in ATM networks may be used by applications via ATM-aware APIs or via IP multicast over ATM. The main advantages of using native ATM for multicast distribution are optimal distribution trees and large data frame sizes (up to 64 kbytes). But in practice, only very few vendors have 622 Mbit/s ATM adapters with native ATM support. Also a pure ATM network is needed between all

[7]The multicast Time-To-Live (TTL) value specifies the number of routers (hops) that multicast traffic is permitted to pass through before expiring on the network. For each router (hop), the original specified TTL is decremented by one (1). When its TTL reaches a value of zero (0), each multicast datagram expires and is no longer forwarded through the network to other subnets.

end systems and the API programmer needs some ATM knowledge (especially about ATM signalling).

2.5.3 Ethernet (Gigabit Ethernet)

In Ethernet-based networks IP multicast groups are mapped to special Ethernet addresses. In older Ethernet switches, packets with multicast addresses are handled as broadcast and flooded to all ports. The high-speed multicast traffic may be distributed via a separate VLAN. Ethernet adapters for end-systems may lack this feature. Modern Ethernet switches are able to restrict multicast traffic to those ports where receivers are located. Two techniques exist (Dörries and Zier, 2001): IGMP snooping works on the IP level and on the registration of IP multicast receivers (via IGMP messages). The GARP[8] multicast registration protocol (GMRP) allows receivers to register Ethernet multicast groups explicitly. IGMP snooping is transparent to IP multicast hosts and more often implemented. Traditionally the payload of Ethernet packets is limited to 1500 bytes, but some vendors also support so called Jumbo frames (up to 9180 bytes) to achieve higher throughput.

Multicast also can be accomplished either by software, by processor-based forwarding mechanisms or by the distribution of multicast packets as broadcast. Thus, the bandwidth available to multicast applications has been restricted to several megabits (Dörries and Zier, 2001). Several technologies like gigabit Ethernet and Asynchronous Transfer Mode (ATM) offer multicast support at high data rates. Although these high-speed technologies have become available to end-users, most users are still using the cheapest Ethernet technology.

This study has concentrated on this technology (Ethernet) as a medium of communication to test WebCOM. The reason Ethernet technology has been chosen for this research is because of its simplicity in providing communication medium for WebCOM without the need for special hardware, configuration and maintenance.

[8]`http://www.alliedtelesyn.co.nz/documentation/at8700/261/pdf/garp.pdf`

2.6 Multicast Protocols

The purpose of this section is to distinguish between two different multicast protocols. Two types of multicast protocols are Routing Multicast Protocol and Reliable Multicast Protocol. Routing Multicast Protocol finds the best path from a sender to a set of receivers and is not concerned with reliability, whereas Reliable Multicast Protocol is concerned with the reliability of data being sent from a sender to set of receivers, but does not seek the best path.

2.6.1 Multicast Routing Protocols

Network routing is concerned with selecting a route from a source to a destination on which the connection will be established. According to Adoud *et al.* (2001), multicasting provides Quality of Service (QoS) guarantees by using multicast routing protocols to find the best path to send the packets. Multicast routing will find a *routing tree* which is rooted from a source and contains all the destinations to be used as a path to send data. Choosing a proper routing tree which maximises the path sharing can significantly reduce the network bandwidth consumption of multicasting.

There are many well-known multicast routing protocols on the Internet today such as Distance Vector Multicast Routing Protocol (DVMRP), Multicast extensions to OSPF (MOSPF), Core Based Tree multicast routing protocol (CBT), and Protocol Independent Multicast (PIM). Different protocols have different design concepts and use different techniques to deliver packets. Appendix D provides brief explanation on some of these Internet protocols.

2.6.2 Reliable Multicast Transport Protocols

Multicast communication, the delivery of a data stream from a single source to multiple destinations in a computer network, has received a great deal of attention in recent years. Much of this interest is due to the development and deployment of IP multicast, which provides best-effort delivery of datagrams to multiple destinations in the

Internet. Applications that can directly take advantage of IP multicast usually involve real-time human interaction and include multimedia conferencing, Internet radio, tele-gaming, and distribution of live and recorded video (McKinley *et al.*, 1999a). For these applications, the unreliable nature of IP multicast communication is acceptable.

However, for many other distributed applications that use multicast communication, the data stream must be reliable, that is, all packets must be delivered in the correct order without loss or corruption. Example applications include distributed parallel processing, bulk distribution of software upgrades, database updates, and distributed operating systems (McKinley *et al.*, 1999a). An efficient reliable multicast protocol can greatly simplify the design of, and improve the performance of, distributed applications. Reliable multicasting also facilitates the use of encryption and compression of the data stream.

In recent years, reliable multicast transport protocols have become an active research area. This problem is fundamentally difficult for three main reasons (McKinley *et al.*, 1999a):

- providing reliability on the data stream requires either positive feedback (ACKs) or negative feedback (NAKs) from each of the receivers; with either method, the processing of control information can simply overwhelm the sender.
- implementing flow control that is acceptable to all receivers is complicated by the presence of heterogenous receivers, disparate communication link capacities, and transient network delays.
- maintaining membership and state information on each receiver is potentially costly in terms of memory and processing overhead. All of these problems are more difficult to solve as the group size increases.

Since the 1980s, a number of reliable multicast transport protocols have emerged. Each of them has its own emphasis and application scope. Several researchers have proposed generic reliable multicast protocols, much as TCP is a generic transport protocol for reliable unicast transmission.

Different applications might have different requirements for reliability. For example, some applications require that delivery obey a total ordering while others do not. Some applications have replicated data, for example in an n-redundant file store, so several members are capable of transmitting a data item while for others all data originates at a single source.

These differences all affect the design of a reliable multicast protocol. Although one could design a protocol for the worst case requirements, such as guaranteeing totally ordered delivery of replicated data from a large number of sources, such an approach results in substantial overhead for applications with more modest requirements. One cannot make a single reliable multicast delivery scheme that optimally meets the functionality, scalability, and efficiency requirements of all applications. This thesis does not discuss reliable multicast transport protocols in more detail, but has chosen one of them to be applied in the study. This study has chosen a reliable multicast protocol that meets the research objectives, with the following criteria:

- the protocol is scalable to deal with the problems of a large group size and in a LAN and WAN environment.
- the protocol should have low-overheads and have a low control traffic.
- no application semantics are included in the transport protocol.
- the prototype should be able to handle heterogeneity of receivers, and to deal with the best and worst receivers on the network.
- the protocol should provide support for multicasting although there is no multicast enabled router available. This feature allows end-users to use a CSCW application without need of a multicast-enabled router.
- the protocol should support multiple-user interaction. This feature enables many users to participate in a discussion in a group.

- the protocol should give the user the ability to act as receiver and sender in duplex mode.
- the protocol should provide a data checking scheme to check the reliability of transmitted data.

There are a number of reliable multicast protocol available:

- *Light-weight Reliable Multicast Protocol (LRMP)*(Liao, 1997)
- *Log-based Receiver Reliable Multicast Protocol (LBRM)*(Holbrook *et al.*, 1995)
- *Reliable Multicast Transport Protocol (RMTP)*(McKinley *et al.*, 1999a)
- *Tree-based Reliable Multicast (TRAM)*(Kandansky *et al.*, 2000)
- *Tree-based Multicast Transport Protocol (TMTP)* (Yavatkar *et al.*, 1995)

Table 2.1: Comparison of Reliable Multicast Protocols

Reliable Multicast Protocol	Category		requires multicast router	platform independent	working simultaneously	support multiple user	scalability	algorithm design			nature of delivery		language/open
	receiver-initiated	sender-initiated						error recovery	flow control	congestion control	ordered	unordered	
LBRM	✓		yes	unknown	unknown	yes	yes	✓			unknown		unknown
RMTP		✓	yes	unknown	unknown	yes	yes		✓		unknown		unknown
TRAM		✓	yes	no	unknown	yes	yes	✓	✓	✓	✓		unknown
TMTP	✓		yes	unknown	unknown	yes	yes	✓	✓		unknown		unknown
LRMP	✓		no	yes	yes	yes	yes	✓	✓	✓	✓		Java/open

Table 2.1 shows the comparison among reliable multicast protocols that might be used in this study. Most of the protocols have features that are needed by the

study. However, the *Light-Weight Reliable Multicast Protocol (LRMP)*(Liao, 1997) has been chosen because this protocol is available for open use, has been written in the Java language, and does not require multicast enabled routers for data transmission. The following section describes the LRMP protocol and Appendix F explains the protocol in more detail. Basically, LRMP has a number of advantages such as:

- reliable and source ordered data delivery service for group communications.
- no prior configuration and no router support are required by providing a random expanding probe scheme that is used for local error recovery.
- it works in heterogeneous network environments and supports multiple data senders.
- provides a flow and congestion control mechanism to fairly share the network bandwidth with other data flows.

There is no specific literature discussing the disadvantages of the LRMP. For more detail information on LRMP, please refer the documents by Liao (1998).

Light-Weight Reliable Multicast Protocol(LRMP)

The LRMP is derived from the Scalable Reliable Multicast Protocol (SRM) (Floyd *et al.*, 1997) but introduces some simplifications and improvements. It was designed to be scalable and lightweight, but was not designed with the tight delivery constraints of real-time traffic. Unlike, SRM, LRMP provides totally ordered packet delivery and rate-based traffic flow control. LRMP has no local recovery. It offers end-to-end reliable and ordered data delivery service to applications. Preliminary tests by Liao (1997) showed that this protocol meets the requirements of file distribution over the MBONE [9]. The detail of this protocol is given in Appendix F.

[9]set of multicast-enabled router

2.7 Web-multicasting

The word "Web-multicasting" in this thesis refer to the implementation of a system that uses multicasting and World Wide Web(WWW) technologies. The next section describes related work on Web-multicasting. It discusses work related to how WebCOM is implemented. It is also shows the position of this research among other related works.

2.7.1 Related Work

This section discusses related work on how WebCOM is implemented in this study. Firstly, a number of Java-based collaboration frameworks have been developed. Many systems, such as JETS (Shirmohammadi *et al.*, 1998) and Promondia (Gall and F.J., 1997), take advantage of the elegance and portability of the Java applet. In such systems, the application components with which users interact are distributed as an applet and a program on the server manages shared state, synchronisation, and dissemination of events. The centralised structure is a consequence of applet security restrictions. Basing the design of a collaborative framework on applets has several advantages. Besides portability, the centralised architecture simplifies handling of many communication and synchronisation issues, compared to distributed solutions. The work in this thesis, WebCOM, uses the methods described above and also extends the technologies to work in a distributed structure.

TANGO (Beca and et. al., 1997) is a Web-based collaboration system that aims to support both synchronised and independent views of related information. Like WebCOM, TANGO uses applets and provides an API to enable the applets to communicate with other programs on the local host or other computers. TANGO also allows the applets to work in a stand-alone fashion or a collaborative fashion. While TANGO is written in Java, it includes APIs for Java, C/C++ and Javascript to provide language independence. A major difference between WebCOM and TANGO is that TANGO uses a centralised server for all communication, whereas WebCOM uses a hybrid architecture.

DISCIPLE (Marsic, 1999) is a Java-based synchronous groupware system that uses the JavaBean component architecture to construct distributed applications. The framework can support any application that is written as a JavaBean, whether designed for collaborating groups or for single users. DISCIPLE uses the Java delegation event model to support dissemination of events from one instance of the application to others. A "collaboration bus" supports all communication services among instances of the application, including reliable multicasting. As with applet-based frameworks, the DISCIPLE framework is very elegant, with well-defined interfaces and composition methods. The target application domain of WebCOM is somewhat more general than that of DISCIPLE, and the primary focus of WebCOM is on communication performance.

Habanero (Chabert *et al.*, 1998) provides state and event synchronisation for multiple copies of an application replicated across different clients. State changes to Java objects are shared by means of a configurable API and an extensible arbitrator manages floor control according to application specific requirements. Existing applets can be ported to Habanero by changing the inheritance of the applet from the standard "Applet" to the "Hablet" class. While WebCOM and Habanero offer similar services, the methods used are quite different. Habanero is built on a client/server architecture, whereas WebCOM is built as a hybrid architecture. In Habanero the server handles the event ordering, while WebCOM uses a LRMP to maintain ordering. Habanero is still based on unicast connections to the server, although the use of multicast has been planned. Finally, applications written for Habanero must be run within the Habanero environment, while WebCOM applications run on any Java-enabled Web browsers.

MultiTel (Fuentes and Troya, 1999) is a Java-based compositional framework for building collaborative applications. MultiTel is structured as two layers, one containing application-dependent components, and the other containing supporting services. An important feature of MultiTel is that the components are run-time configurable and can accommodate varying qualities of service. MultiTel provides a convenient framework for building applications by abstracting away from the underlying details. As noted by the authors, however, use of Java can produce performance problems. Although WebCOM provides similar capabilities to MultiTel, the goals of the projects are complementary. Whereas MultiTel focuses on functionality and ease of use, WebCOM is focused on performance issues.

MASH (McCanne *et al.*, 1997) addresses the communication aspects of collaborative applications and supports high-bandwidth data transfers such as streaming media. MASH can be used to interconnect different components or extend existing components. The MASH project uses the SRM protocol (Floyd *et al.*, 1997) for reliable multicasting, a flexible proxy server for layered transmission and transcoding (Chawathe *et al.*, 1998), and a component for HTML distribution called MASHCast. Many of the MASH components are legacy systems written in C/C++. WebCOM uses the LRMP protocol (Liao, 1997) for reliable multicasting and is written using the Java language.

Pavilion (McKinley *et al.*, 1999a) is a middleware framework that supports the development of collaborative Web-based applications. Pavilion enables a developer to construct new collaborative applications by inheriting and extending the default functionality of Pavilion. A key-principle followed in Pavilion is integration of existing applications, including various types of browsers and data-specific interfaces. In Pavilion several components have been used including the proxy server, Web browser interface, and reliable multicast protocol. Pavilion used WBRM (Web-Based Reliable Multicast) as a reliable multicast protocol, while WebCOM used LRMP as a reliable multicast protocol. Pavilion and WebCOM are written in Java to provide language independence. WebCOM focuses on communication issues and a major difference between the projects is that Pavilion uses a distributed architecture for all communication, whereas WebCOM uses a hybrid architecture.

In summary, there are many different ways to design frameworks to support collaborative applications, and the research community is actively pursuing numerous alternatives. WebCOM exhibits similarities, and differences, with several systems. The primary goal of WebCOM was to enhance a communication framework and study per-

Table 2.2: Research Area for WebCOM

Framework	Year	Type	Architecture	Usage	Multicast	Protocol
Promondia	1997	synchronous	client/server	collaborative	none	none
JETS	1997	synchronous	client/server	up to two clients	none	none
MASH	1997	synchronous	distributed	collaborative	yes	SRM
Habanero	1998	synchronous	client/server	stand alone	none	ordering handle by server
TANGO	1997	synchronous	client/server	stand alone / collaborative	none	none
DISCIPLE	1999	synchronous	distributed	stand alone / collaborative	none	none
MultiTel	1999	synchronous	distributed	stand alone/ collaborative	none	none
Pavilion	1999	synchronous	distributed	stand alone / collaborative	multicasts URLs and web resource from leader's to all participants	distributed leadership protocol-ordering
JASMINE	2000	synchronous	client/server	none	none	none
WebCom	2003	asynch/synch	hybrid - client/ server & peer-to-peer	more than two clients	multicast web resources from server to client and from client to client	LRMP

formance issues arising in collaborative communication systems. The related work that has been discussed can be viewed in Table 2.2. In this table, the studies have been categorised into five features as below:

- **Communication Type**

  There are two types of communication: synchronous and asynchronous. Synchronous communication involves two or more parties communicating at the same time, while asynchronous communication involves two or more parties communicate at different times.

- **Architecture**

  There are five types of network architecture: centralised (client/server), replicated, peer-to-peer, distributed, and hybrid. Some of these type are explained in Appendix B.

- **Usage**

  The ability of a framework to work stand-alone or for two or more users indicates the scalability of that framework. In other words, the more users that can be supported by the framework, the more scalable the framework.

- **Use of Multicasting**

  There are three methods of communications: unicasting, broadcasting and multicasting. All of these methods has been discussed in Chapter 2, Section 2.4. Most of the applications use unicasting as the communication method because it is easier to develop and maintain. It is very rare to find an application developed on the basis of multicasting because this method is relatively new.

- **Protocols**

  The protocol is used to ensure reliability of data being transferred in the network. Without the protocol, the reliability of the data is not guaranteed.

2.8 Issues of Performance

Multicasting is a kind of group communications which requires simultaneous transmission of messages from a source to a group of destinations. This section describes approaches to improve performance from different authors and research fields.

2.8.1 Approaches to improve performance

There have been many efforts to improve the performance of applications on the network. Each method proposed has pointed to a specific problem or issue. A number of related works has been found in the literature:

- Multicasting is able to provide better responsiveness with some modifications of communication method such as separating different data types in different channels (Parr and Curran, 1999; Keshav and Paul, 1998).

- Balter *et al.* (1995) have given three standard architectures for CSCW: central architecture, replicated architecture, and hybrid architecture. Architecture scheme selection can impact the overall multicasting performance, for example response time and delay.

- Parnes *et al.* (1996) and Liao (1997) have used a Web-proxy to intercept participant's browser requests.
- Coulouris (2001) has extended the WWW architecture, using a proxy server and cache to increase the availability and performance of the services by reducing the load on the wide-area network and Web servers.
- Rodriguez *et al.* (1998) have used multicasting and Web cache to improve WWW performance.
- Gemmell *et al.* (1998) pointed out that data types also have an impact on the overall multicasting performance. They have focused on unreliable transmission where some data required timely delivery and other data required reliable delivery.
- Zabele *et al.* (1994) have used reliable and unreliable multicasting protocols to increase the information reliability. The protocols used depend on whether data requires reliable or unreliable transmission.
- Matta and Eltoweissy (1997) have used a multicast routing protocol to achieve stable performance in real-time CSCW applications.
- Shin (1997), Shin and Lee (1998), and Shin *et al.* (1998) have used a protocol to support real-time streaming on the Web, using multicasting to achieve higher performance.
- Keshav and Paul (1998) have used separation of multicasting data and control to get higher performance.
- Parr and Curran (1999) have used different channels to transmit different data types in multicasting.

2.8.2 Performance Metric

Most of the authors took two important metrics in their study: throughput and overhead (Hong *et al.*, 1999), packet loss and round-trip loss (Borella *et al.*, 1998), throughput and latency (Ahmed *et al.*, 2002), and delay and packet loss (Barford and Crovella, 1999). Most authors have measured performance based on response time and latency (Karger *et al.*, 1999; Arlitt *et al.*, 2000; Liu *et al.*, 2001), throughput and packet loss (Nandagopal *et al.*, 2004) . However, the performance metrics used for experiment in this thesis are response time, throughput, and packet loss. The metrics definitions used in this thesis are:

- Response time:
  The elapsed time between the end of an inquiry or demand on a computer system and the beginning of a response, for example, the length of the time between an indication of the end of an enquiry and the display of the first character of the response at a user terminal. Or in other words, the time it takes a system to react to a given input. The response time includes the transmission time, and the processing time.

- Packet Loss:
  Number of packets lost during transmission between nodes in the network. Packet loss is measured either in percentage or amount loss of data.

- Throughput:
  The amount of data transferred in one direction over a link divided by the time taken to transfer it. It is usually expressed in bits or bytes per second. Throughput is calculated as the data size divided by the transfer time (Wolski, 1998).

$$\textit{throughput} = \textit{data size} /(\textit{data transfer time})$$

2.8.3 Accuracy

There are two terms that used for referring to time clock: *stability* and *accuracy.* The term *stability* refers to how well a clock ticks at a constant frequency (Mills, 1990), which is not discussed in detail in this thesis. This thesis will assume that the clock systems involved in this research are stable. The term *accuracy* refers to how well its time value compares to a reference clock (Mills, 1990), which is considered in this thesis as an important part in the experimental phase. Note that the accuracy of a clock is not its resolution. Sun Sparcstations have μsec resolution (the UNIX system call gettimeofday() returns a μsec value), however their accuracy is tens of msecs (Dietz *et al.*, 1995). Another examples are:

- Java has millisecond accuracy with a measuring convention that is transparent to the user (because it is internal to the Date class).
- VRML has microsecond accuracy with a measuring convention specified from Jan 1, 1970, 00:00:00 GMT.

The experiment in this thesis used Java as the development language. For time accuracy, this study has used *Java: Class java.util.Date*[10]. The class Date represents a specific instant in time, with millisecond precision. The Date class is intended to reflect coordinated universal time (UTC).

2.8.4 Synchronisation

Workstations in a local area network environment must provide a time service that is synchronised with the other clocks on the LAN. The clocks of all hosts in a session could be synchronised, by using a protocol such as the Network Time Protocol (NTP) or by Global Positioning System (GPS) receivers. It is essential for any measurement involving time to calculate the delay accurately, that is, the clock must be synchronised for each involved node. Clock synchronisation can be achieved in several ways, such as software synchronisation, hardware synchronisation, or a combination of the two. The work in this thesis has used software to synchronise the clock. Further discussion on synchronisation and tools is given in Appendix G.

[10]`http://java.sun.com/products/jdk/1.1/docs/api/java.util.Date.html`

2.8.5 Reliability

According to ANSI/IEEE (1991) the definition of reliability is "The probability of failure-free software operation for a specified period of time in a specified environment". This is widely used but it is not the only definition. Another commonly used way to express reliability is Mean Time To Failure (MTTF). Reliability is one characteristic of software quality that often is considered as important. Even if there is no danger involved when a program fails, the user will probably not tolerate this happening too frequently. Reliability is important to indicate the status of certain system working in a correct order or working as it supposed to. The system is called reliable if it fulfills the requirements of that system without error (fault) and failure. Appendix H explains a number of ways to measure the reliability of a system.

2.9 Technologies Support

2.9.1 Java Technology

This section describes briefly the Java language used for developing LRMP, WebCOM, and prototype system in this study. Java has some weakness such as run-time performance, but this issue can be solved by tuning the Java programming as explained in detail by Shirazi (2000). Some of the advantages of Java are:

- platform-independence
- memory management
- powerful exception checking
- built-in multithreading
- dynamic resource loading
- security checks

The specification of the Java language can be found from `http://java.com.sun/`.

2.10 Summary

This study has focused on providing communications to support CSCW applications. Although many CSCW applications have been developed and are being used, the demands of users are increased on the available technology changes. This study has identified a number of issues in the CSCW applications such as availability, portability, scalability and performance. World Wide Web (WWW) and multicasting are the two technologies that has been chosen by this study to improve the efficiency of CSCW applications. These technologies have been chosen because they can provide a solution for the above issues in CSCW. The WWW offers a globally accessible platform independent infrastructure, well integrated with others environment and technology such as Java and CGI. Multicasting gives advantages of scalability and performance. These technologies have been used to develop WebCOM as a communication framework for CSCW applications.

The WebCOM has enhanced previous related studies to enable the implementation of CSCW applications. The common method that all of the works used is the implementation of application using applets on the Internet through Web browser. The modification and enhancement of WebCOM from previous works are:

- enhance the application architecture from central to hybrid. This refers to JETS (Shirmohammadi *et al.*, 1998), Promondia (Gall and F.J., 1997), and TANGO (Beca and et. al., 1997)
- primary focus of WebCOM is on the communication performance. This refers to DISCIPLE (Marsic, 1999), and MultiTel (Fuentes and Troya, 1999)
- running in collaborative environment such as CSCW application. This refers to Habanero (Chabert *et al.*, 1998)
- implement reliable multicast protocol, LRMP, in WebCOM to provide reliability and efficiency. This refers to Habanero (Chabert *et al.*, 1998) and Pavilion (McKinley *et al.*, 1999a).

- implement WebCOM using the Java language. This refers to MASH (McCanne *et al.*, 1997).
- implement for asynchronous and synchronous CSCW applications. This refers to all above works since all other implementation are synchronous.

Although there are various methods of implementation of IP Multicast in the Internet such as Ethernet, ATM, and IP router, this study has chosen Ethernet implementation because many end users have this technology and it is cheap compared to the others. Although it has limitations such as limited data rate transfer, this does not effect the objective of the implementation of IP multicast itself.

The work in this thesis tests whether WebCOM can support a CSCW application, and provide an efficient environment to CSCW applications, by determining the operation of CSCW tasks. The efficiency of the WebCOM will be measured by measuring and analysing performance of the prototype. The study in this thesis has developed a prototype of CSCW application using the Java language. The prototype will aim to provide as many CSCW application components as possible to make its conclusions applicable to CSCW applications other than the prototype application.

Chapter 3

WebCOM and Prototype Design

3.1 Introduction

This chapter describes WebCOM design and a prototype of CSCW applications. The design includes architectures, protocols, communication flows, and prototype modules.

3.2 Aim of Proposed Model

The purpose of work in this study, as described in Chapter 1, is to develop the WebCOM System, which includes:

- implementation of WebCOM on hybrid architecture (peer-to-peer and client/server).
- integration of a reliable multicast protocol (RMP) for the WebCOM System.
- an implementation of a prototype on the WebCOM System.
- extending the scalability of existing technology to manage a large user group using multicasting with WebCOM System.

Table 3.1: Existing Implementations on Web

Framework	Year	Type	Architecture	Usage	Multicast	Protocol	Environ-ment	Langu-age
Promondia	1997	synchronous	client/server centralise	collaborative	none	none	none	Java/Applet
JETS	1997	synchronous	client/server centralise	up to two clients	none	none	none	Java/Applet
MASH	1997	synchronous	distributed	collaborative	yes	SRM	WWW	C++
Habanero	1998	synchronous	client/server	stand alone	none	ordering handle by server	WWW	Java/C
TANGO	1997	synchronous	client/server	stand alone / collaborative	none	none	WWW	Java/APIs
DISCIPLE	1999	synchronous	distributed	stand alone / collaborative	none	none	WWW	Java/ JavaBean
MultiTel	1999	synchronous	distributed	stand alone/ collaborative	none	none	WWW	Java/RMI
Pavilion	1999	synchronous	distributed	stand alone / collaborative	multicasts URLs and web resource from leader's to all participants	distributed leadership protocol-ordering	WWW	Java/APIs
JASMINE	2000	synchronous	client/server	none	none	none	WWW	Java/Applet

3.3 Existing Models: Limitations and Strengths

Table 3.3 shows existing frameworks using Java applets in a Web environment; this has been discussed in Chapter 2, Section 2.7. The related work that has been discussed is summarised in Table 3.3. The features chosen to categorise the framework are based on use, architecture and implementation aspects that arise in WebCOM. The similarities and differences among the frameworks are identified from these features, and elements of novelty of this study are identified from these features. The studies have been using the six main features below:

- **Communication Type**

  There are two type of communication: synchronous and asynchronous. Synchronous communication involves two or more parties communicating at the same time, while asynchronous communication involves two or more parties communicate at different times. This is important in CSCW because from Table 3.3 most of the implementations use synchronous mode.

- **Architecture**

  There are five types of network architecture: centralised (client/server), replicated, distributed, peer-to-peer and hybrid. These types have been explained in Appendix B. Most studies used client/server and distributed architectures. JETS (Shirmohammadi *et al.*, 1998), Promondia (Gall and F.J., 1997), TANGO (Beca and et. al., 1997), use client/server architecture; while DISCIPLE (Marsic, 1999), MultiTel (Fuentes and Troya, 1999), and Pavilion (McKinley *et al.*, 1999a) use distributed architecture.

- **Usage**

  The ability of a framework to work stand-alone or to work for two or more users indicates the scalability of that framework. In other words, the more users that can be supported by the framework, the more scalable the framework. Scalability is an important features in CSCW. Most studies offer stand alone and/or collaborative except JETS (Shirmohammadi *et al.*, 1998) and Habanero (Chabert *et al.*, 1998) which only work as stand alone.

- **Multicasting**

  There are three methods of communications: unicasting, broadcasting and multicasting. All of these methods have been discussed in Chapter 2, Section 2.4. Most of the applications use unicasting as the communication method, because it is easier to develop and to maintain. There are few applications developed on the basis of the relatively new multicasting method. Pavilion (McKinley *et al.*, 1999a), MASH (McCanne *et al.*, 1997), and Habanero (Chabert *et al.*, 1998) are examples which use multicast protocols in their work.

- **Language**

  There are many languages used, for example Java, C, C++, and Javascript. Each language has its own advantages compared to others. For example, the Java language provides an easy way to connect an application to the Internet, and can work on any platform. Normally, choice of the language depends on users requirements. Many works have used Java applets in their implementation on a Web environment, for example, JETS (Shirmohammadi *et al.*, 1998), Promondia (Gall and F.J., 1997), and Habanero (Chabert *et al.*, 1998). Some of them have used Java with other languages such as TANGO (Beca and et. al., 1997), which was written using Java, C/C++ and Javascript to provide language independence; while DISCIPLE (Marsic, 1999) has written using Java/JavaBean.

- **Protocols**

  The protocol is used to ensure reliability of data being transferred in the network. Without the protocol, the reliability of the data is not guaranteed. Two works have used multicasting in their application and reliable multicast protocol in their implementation: MASH (McCanne *et al.*, 1997) and Pavilion (McKinley *et al.*, 1999a).

Looking at the current implementation of Java applets in a Web environment as shown in Table 3.3, a number of limitations/strengths are identified:

- most of the frameworks run in synchronous mode. Although synchronous mode is the communication mode used in most previous CSCW applications, a generic CSCW application should support both modes of interaction: asynchronous and synchronous.

- unicasting is a standard method and is adopted by many authors to develop their applications/model. This method is very well known and does not need to be described here. However, currently multicasting has received more attention with many researchers beginning to investigate the potential of multicasting to provide an alternative method of communication. In the CSCW area, few studies have used multicasting. As reported by Maihöfer (2000), multicasting might save network bandwidth and offers an efficient communication technique for a group communication purpose. Moreover, multicasting provides scalability and speed performance (Maihöfer, 2000);

- most of the frameworks use client/server or distributed architecture, and each architecture has its own advantages and disadvantages. Most of the works preferred to use either one of these architecture.

3.4 Comparison of WebCOM with Existing Framework

Table 3.2: Comparison of WebCOM with Existing Model/Framework

Framework	Year	Type	Architecture	Usage	Multicast	Protocol	Environment	Language
Promondia	1997	synchronous	client/server centralise	collaborative	none	none	none	Java/Applet
JETS	1997	synchronous	client/server centralise	up to two clients	none	none	none	Java/Applet
MASH	1997	synchronous	distributed	collaborative	yes	SRM	WWW	C++
Habanero	1998	synchronous	client/server	stand alone	none	ordering handle by server	WWW	Java/C
TANGO	1997	synchronous	client/server	stand alone / collaborative	none	none	WWW	Java/APIs
DISCIPLE	1999	synchronous	distributed	stand alone / collaborative	none	none	WWW	Java/ JavaBean
MultiTel	1999	synchronous	distributed	stand alone/ collaborative	none	none	WWW	Java/RMI
Pavilion	1999	synchronous	distributed	stand alone / collaborative	multicasts URLs and web resource from leader's to all participants	distributed leadership protocol-ordering	WWW	Java/APIs
JASMINE	2000	synchronous	client/server	none	none	none	WWW	Java/Applet
WebCom	2003	asynch/ synch	hybrid - client/ server & peer-to-peer	more than two clients	multicast web resources from server to client and from client to client	LRMP	WWW	Java/Applet/ APIs

The work of this thesis proposes WebCOM. WebCOM is a communication framework that uses the Java language to enable it work at any platform in heterogeneous environments. WebCOM proposes to overcome the limitations that have been described (Section 3.3) by:

- developing a framework that supports both communication modes, namely, asynchronous and synchronous.

- adapting multicasting as a main method of communication in WebCOM.

- using hybrid architecture to get full advantage from the components of the hybrid architecture.
- adapting reliable multicast protocol to achieve good QoS.

The design of WebCOM in relation to other frameworks is shown in Table 3.2.

The work in this thesis uses Java APIs to extend the capabilities of multicasting and offers a reliable scheme for a prototype to work collaboratively in World Wide Web environment. Java provides a standard mechanism for multicasting where a source component is able to generate an event and send it to a set of listeners (receivers).

3.5 Architecture of WebCOM

WebCOM has used a hybrid architecture to give advantages to group communication compared to the client/server architecture. The client-server architecture is not suited to a large size of group environment because:

- processing cost of a server increases with the number of users accessing the server.
- response time increases with the number of users accessing the server.
- user cannot communicate directly to other users unless they are connected to a the same server. If the server is down, the communication cannot be established.
- every user needs to have a connection to a server in order to communicate with other users.

The above problems as a group requirement can be solved by proposing WebCOM as the communication framework. The objectives of WebCOM architecture are to give advantages to the participant and the server by eliminating the server dependent processes, decreasing server load and decreasing network traffic. For example, users are not totally dependent on a Web server if a Web server is down or fails.

This study has merged the client/server architecture with the peer-to-peer architecture to produce a three-tiered architecture for WebCOM as shown in Figure 3.1. The

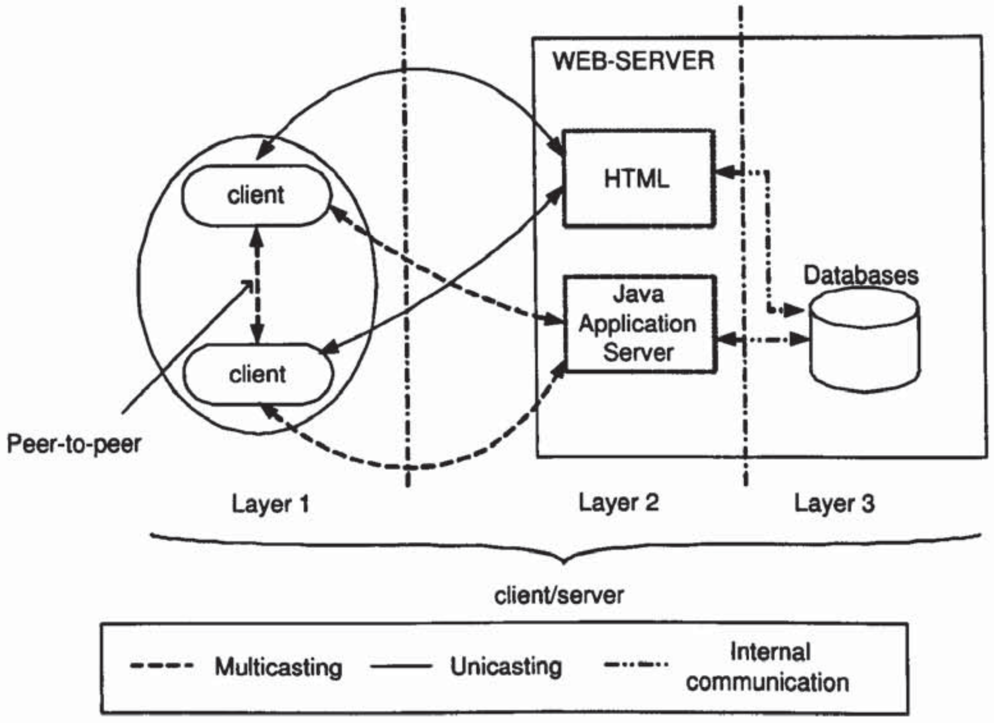


Figure 3.1: Hybrid Architecture for WebCOM

three-tiered architecture is divided into three layers. All clients are categorised into the first layer, Java Application Server(JAS) and HTML (Home page of the prototype system) are categorised into the second layer, and the databases is categorised into the third layer. The first layer is located on a local or remote network, and the second and the third layers are located on a local or remote Web Server. The client/server architecture is implemented for client-Web server communication and the peer-to-peer architecture is implemented for client-client communication.

Two types of communications are used in the hybrid architecture i.e. unicasting and multicasting. Unicasting is used for client-HTML(web server components) communication in which a client communicates with the web server to download java applets. When clients first use the prototype system, all clients need to download a java applet by accessing the server HTML pages using unicasting. The communication between HTML-databases and JAS-databases uses internal communication on the server.

Multicasting is used for client-client communication and clients-JAS communication. After the clients have downloaded the java applet from a web server, all activities in the system use multicasting as the medium of communication either between clients or with JAS. The activities includes:

- updating data in databases through JAS by clients,
- communication between client and client,
- communication between client and a JAS,
- joining a group in the system, and
- communication between clients using shared application which created within JAS or self-created by an authorised client.

A communication made by a shared application that has been created by an authorised client rather than delivered from the server, operates independently from the JAS. In other word, all activities and data are not recorded and traced by the JAS. Furthermore in this thesis, data distributed among clients and not recorded in the databases limits development of the system, and is seen as an area for further research.

The hybrid architecture offers the advantages of client/server and peer-to-peer architectures in that it is central point independent, provides easy management, and reduces bottlenecks. The benefits for this extended architecture are:

- processing cost of server is reduced even though the number of users using WebCOM has increased.
- response time is no longer dependent on the server.
- users can communicate directly with other users without establishing the connection to the server if they have the same copy of an application as downloaded from the server.

- every user does not need to have a connection to a server in order to communicate with other users in a group.

From Figure 3.1, Layers 1, 2 and 3 are from client/server architecture, whereas internally layer 1 is from peer-to-peer architecture. A difference between the WebCOM architecture and client/server architecture is that multicasting has been used in the WebCOM as a communication method for a group interaction, which is shown shaded in Figure 3.1. The operation of the architecture used by WebCOM described in detail in the following subsections.

3.5.1 Layer 1

The first layer handles all activities on the client side. In this layer, the activities are divided into three parts:

- ***Communication with Web Server***
  A client communicates using the unicasting method with a Web Server to access Web pages that contain both information and a directed link to the WebCOM System. The WebCOM System starts by downloading a WebCOM Client from the Web Server. All communications are conducted using the multicast method between a WebCOM Client on the client side and WebCOM Server on the server side after a WebCOM Client has been downloaded.

- ***Communication with WebCOM Server***
  A WebCOM Client, on the client side, communicates using the multicast method, with a WebCOM Server on the server side on the following occasions:

  1. when a WebCOM Client requests information on a particular application from WebCOM Server. For example, a WebCOM Client (BBS) requests a list of events that are going to happen in the WebCOM System.
  2. when a WebCOM Client receives information on particular applications that are going to happen in the WebCOM system from the WebCOM Server.

For example, a WebCOM Server sends a list of events that are going to happen in the WebCOM System in response the WebCOM Clients.

3. when a WebCOM Client sends information on users/group and information to control adding, deleting, and updating of data to the WebCOM Server.
4. when a WebCOM Client sends information on authorization to the WebCOM Server.
5. when a WebCOM Client receives the status of authorization from the WebCOM Server.

- ***Communication between WebCOM Clients***

  All WebCOM Clients are able to communicate using the multicasting method among themselves as required for particular applications.

In conclusion, the multicasting communication method is used between WebCOM Server and WebCOM Clients, and among WebCOM Clients. The unicasting communication method is used between client and Web server at the initial stage i.e. while accessing Web pages that contain information and a directed link to the WebCOM System. The WebCOM System is started by downloading a WebCOM Client from the Web Server, and a WebCOM Server which is always running on a Web Server.

3.5.2 Layer 2

The second layer consists of a WebCOM Server that act as a Java Applet Server. The WebCOM Server is responsible for any client's requests, and for communications with the database and file system when needed. There are three types of communication involved:

- ***Communication with WebCOM Clients***

  A WebCOM Server communicates with WebCOM Clients on the following occasions:

  1. **when a WebCOM Server receives:**

  - a request for user authorisation from WebCOM Clients
  - information to control adding, removing, and updating of data from the WebCOM Clients. Only authorised users are allowed to perform those processes and allowed to operate through WebCOM Clients.

2. **when a WebCOM Server sends**:
   - the authorisation status to the WebCOM Clients in response to a client request.
   - information on particular application to WebCOM Clients is response to client request. For example, a WebCOM Server sends a list of events that going to happen in the WebCOM System to the WebCOM Clients (BBS).
   - status for particular processes such as the processes of adding, deleting, and updating data to WebCOM Clients which are initiated by the WebCOM Server for every process.

- ***Communication with Databases***

  A WebCOM Server communicates with databases on the following occasions:

  1. when a WebCOM Server checks the user's authorisation by comparing the received information from the WebCOM Client with stored information in the databases.
  2. when a WebCOM Server requests the information on particular application in response to a client request. For example, a WebCOM Server requests a list of events that are going to happen in the WebCOM System for a WebCOM Clients (BBS).
  3. when a WebCOM Server is storing, adding and updating valid information on WebCOM users.

3.5.3 Layer 3

The third layer contains databases which store information about clients, groups, moderator, and server. The data types for each database on the WebCOM System are described in Appendix I.10.2. The databases provide information in response to any request from the WebCOM Server. The databases communicate directly to the WebCOM Server on the following occasions:

1. when a database receives a request of checking the user's authorisation.
2. when a database sends information on a particular application (WebCOM Client BBS).
3. when a database is storing, adding and updating valid information of WebCOM users.

3.6 Design of WebCOM

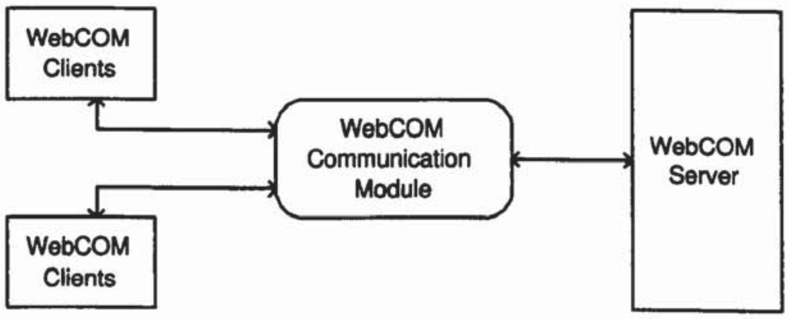


Figure 3.2: Design of WebCOM Architecture

The implementation architecture design of WebCOM is adapted from the design of JASMINE (El-Saddik *et al.*, 2000) because :

- the objectives of JASMINE are a subset of objectives of this thesis intended to solve problems of platforms-independence and working in a heterogenous environment, and JASMINE used the Java language in its implementation.

- JASMINE used applets in its design
- it meets the requirements of WebCOM

Figure 3.2 illustrates the overall concept of WebCOM. The principal idea of the WebCOM implementation is that user events occurring through interaction with the GUI of an applet can be caught, distributed, and reconstructed, hence allowing for Java applets to be shared transparently. This form of collaboration is supported as long as a group session takes place, and enables users to interact in real-time, working remotely as a team without caring about low-level issues. The WebCOM client listens to all events occurring in the graphical user interface of the applet and transmits these events to all other participants in order to be reconstructed there. The WebCOM client captures both AWT-based and Swing-based events. After capture, the event is sent to the communication module (WebCOM Communication Module), where the event is forwarded to all other participants in the session. In this situation, the WebCOM Server is not involved in the communication.

WebCOM Server act as daemon, listens to all incoming events/request from WebCOM clients through communication module. The events/requests involved in the WebCOM system are authentication process, list of events, and user/group registration and updating. In summary, WebCOM itself is divided into three parts as shown in Figure 3.2, discussed in more details in the next sections:

- WebCOM Client
- WebCOM Server
- WebCOM Communication Module

3.6.1 WebCOM Client

The WebCOM Client can be seen as a component adapter as shown in Figure 3.3. Every event occurring at the graphical user interface of the application is sent to listener

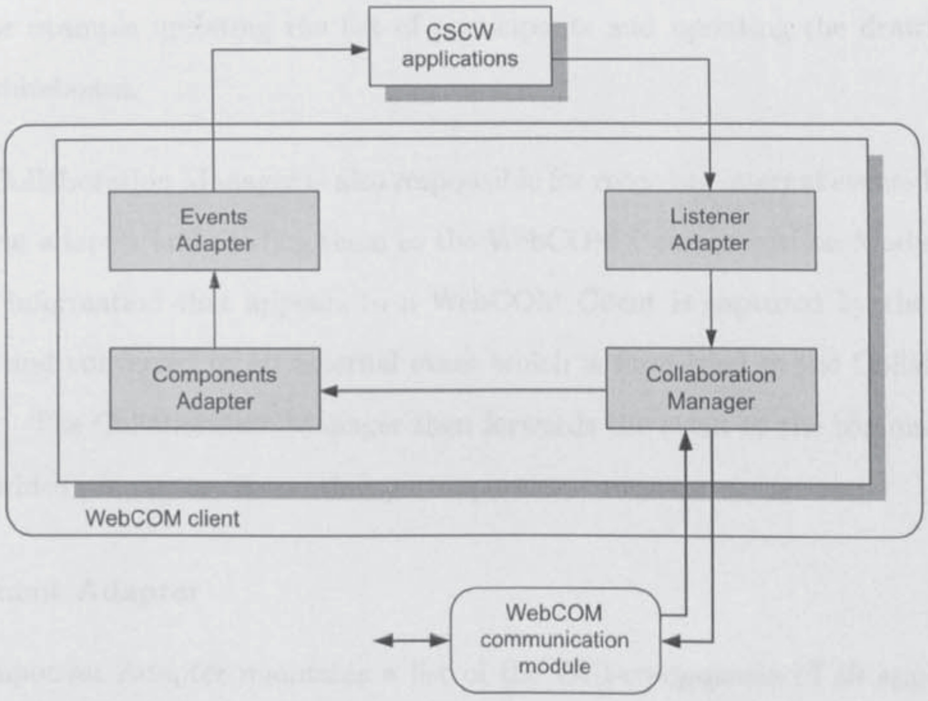


Figure 3.3: Architecture of WebCOM Client

adapters, which then send the events via the Communication Module to the WebCOM-Server and/or another WebCOM Client. The client is a Java applet application, which consists of the following components: *Collaboration Manager, Component Adapter, Listener Adapter, and Event Adapter.*

Collaboration Manager

The Collaboration Manager provides end-users with a graphical interface offering options such as joining a session, sharing a workspace and sharing a chatting board with other participants. Two types of events exist in a WebCOM client, internal events that come from component adapters in the WebCOM client and external events that come from the communication module. The Collaboration Manager handles both internal and external events.

The Collaboration Manager is responsible for dispatching external events coming from the communication module and forwarding them to the component adapter. The Component adapter uses the events for further processing or execution in the WebCOM client, for example updating the list of participants and updating the drawing of a shared whiteboard.

The Collaboration Manager is also responsible for receiving internal events from the component adapter and sending them to the WebCOM Communication Module. Any updated information that appears in a WebCOM Client is captured by the listener adapter, and converted to an external event which is forwarded to the Collaboration Manager. The Collaboration Manager then forwards the event to the communication module which propagates it to other participants.

Component Adapter

The Component Adapter maintains a list of the GUI-components of all applications and applets that running on the client. This list is created with the help of the *java.awt.Container class.* With the help of the main window of an application, a list of the GUI components in the application can directly be created. Therefore, the main window of an application loaded by the Collaboration Manager is registered by the Component Adapter. However, Java applets do not use stand-alone windows. They are an extension of the class java.applet.Applet and thus of java.awt.Panel. Hence, applets can be easily placed into a window, which can then be registered as the main window for the applet. All these registrations are done at the Component Adapter.

After the registration is completed, a list of all Swing and/or AWT-components within the loaded application/applet is created. This task is done in the same order on each client, so that a component has the same reference identification at all clients. These references are used to point to specific components, which are the source of the events generated internally and the recipient of the events generated externally. With the help of the references, the recipient of an incoming event is located and the event is reconstructed on each client, as if it occurred locally. This allows shared space application between clients.

Listener Adapter

The Listener Adapter implements several AWT listeners, which listen to MouseEvent and KeyEvent for all AWT-components. For these components the Listener Adapter listens to ItemEvent and ActionEvent. When an event occurs on the GUI of the application, the Listener Adapter catches it, converts it to an external event, and forwards it to the Collaboration Manager. The Collaboration Manager in turn sends this event to the WebCOM Communication Module, which propagates the event to all other participants.

Event Adapter

The Event Adapter works in the opposite way to the Listener Adapter: it converts incoming external events to AWT events, which then can be processed locally.

Data Flow

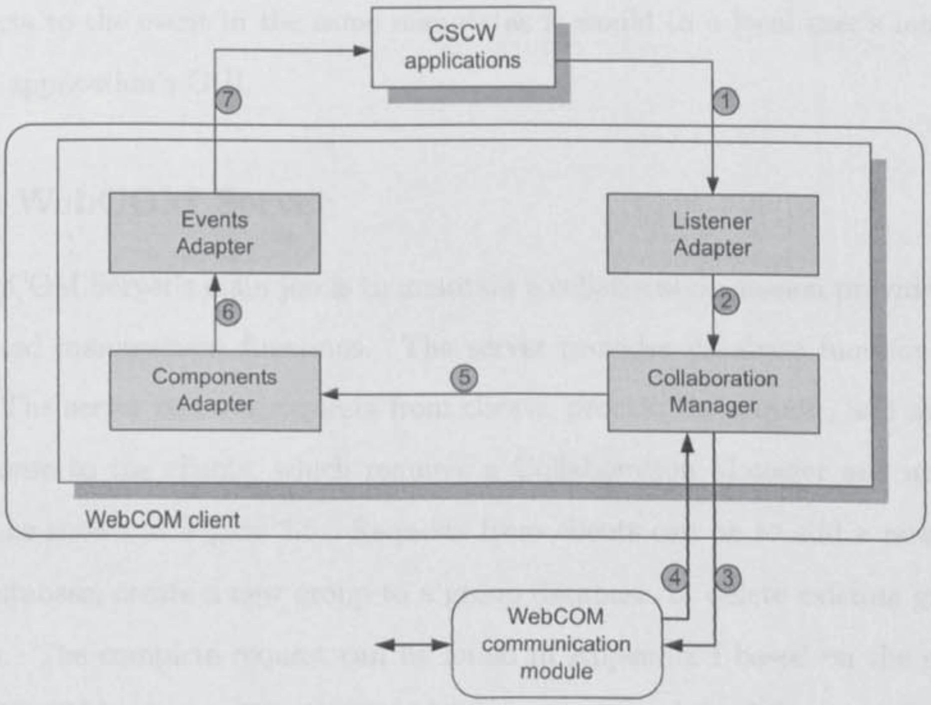


Figure 3.4: WebCOM Dataflow

Figure 3.4 summarises the client side's architecture through a data flow diagram. It shows the overall event circulation of the system. There are two main data paths in the system: the first path is labeled with numbers 1, 2 and 3. This path is used to send the internal AWT events to the WebCOM Communication Module, and it works as follows: any Event that has occurred in a Java applet application is caught by the Listener Adapter. The Listener Adapter first tests whether the event is an external or an internal event. It then sends only the internal events, which were not received from other clients, to the Collaboration Manager, which in turn sends the events to the WebCOM Communication Module. Via the second data path shown in Figure 3.4 with numbers 4, 5, 6 and 7, the external AWT events received from the WebCOM Communication Module are captured by the Collaboration Manager and the Component Adapter in order to reconstruct the event locally. After receiving the remote event, the Component Adapter extracts the information about its target component and sends this information together with the events to the Event Adapter. The Event Adapter converts the event to normal AWT events and sends them to the application, which then reacts to the event in the same manner as it would to a local user's interaction with the application's GUI.

3.6.2 WebCOM Server

The WebCOM Server's main job is to maintain a collaboration session providing moderation and management functions. The server provides database functions to the clients. The server receives requests from clients, process the request, and send back the response to the clients, which requires a Collaboration Manager and an Events Adapter as shown in Figure 3.5. Requests from clients can be to add a new user to a user database, create a new group to a group database, or delete existing group application. The complete request can be found in Appendix I based on the activities in the WebCOM system. The WebCOM Server consists of the following components: *Collaboration Manager* and *Event Adapter.*

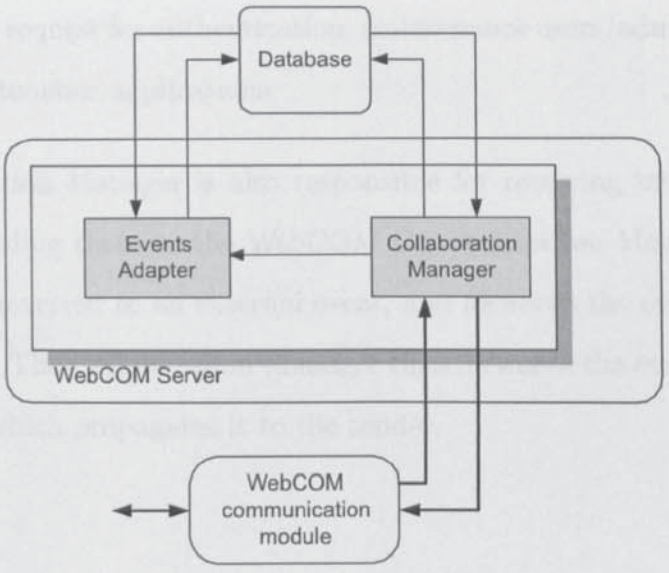


Figure 3.5: Architecture of WebCOM Server

Collaboration Manager

The Collaboration Manager is the main component on the server side and processes all requests from clients such as user authentication and interaction with database. The Collaboration Manager at the server side is different from the Collaboration Manager in a WebCOM Client because the internal interaction for the Collaboration Manager in the WebCOM Server is between database and Events Adapter; while the internal interaction for the Collaboration Manager in the WebCOM Client is between CSCW applications and WebCOM client's adapters. The external interaction is the same for both Collaboration Managers.

Two types of event exist in the WebCOM Server, internal events that comes from component adapters in WebCOM Server and external events that comes from the communication module. The Collaboration Manager handles both internal and external events.

The Collaboration Manager is responsible for dispatching external events coming from the communication module and forwarding them to the component adapter. The Component adapter uses the events for further processing or execution in the WebCOM client, for example request for authentication, maintenance users/administrators/groups records, and maintenance applications.

The Collaboration Manager is also responsible for receiving internal events from databases and sending them to the WebCOM Communication Module. Response of any request are converted to an external event, and forwards the event to the Collaboration Manager. The Collaboration Manager then forwards the event to the communication module which propagates it to the sender.

Event Adapter

The Event Adapter converts incoming external events from a client which can be processed locally.

3.6.3 WebCOM Communication Module

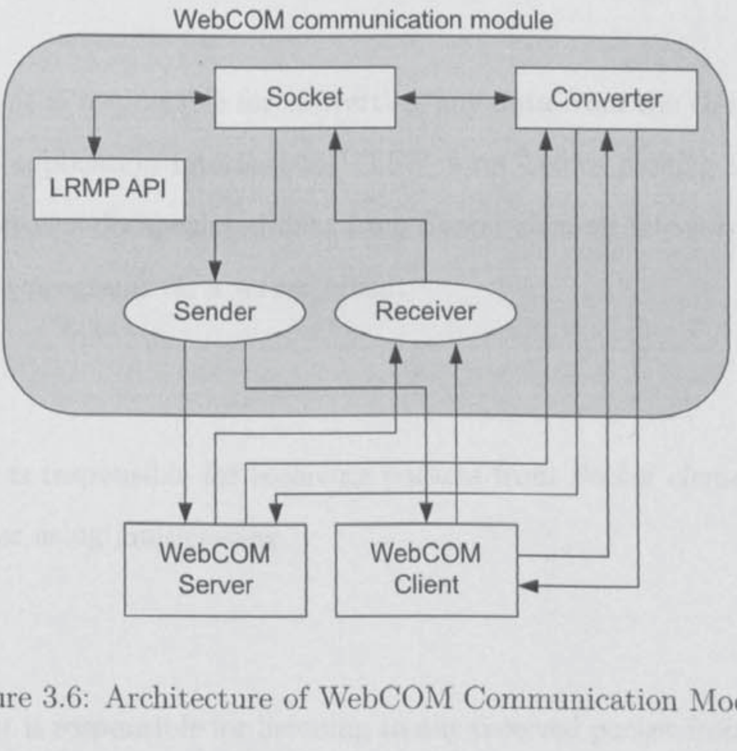


Figure 3.6: Architecture of WebCOM Communication Module

Figure 3.6 shows the WebCOM Communication Module architecture. This module receives incoming and outgoing data/packets from WebCOM Server and Web-

COM Client. The WebCOM Communication Module consists of five elements, namely, LRMP API, *Sockets*, *Converter*, *Sender*, and *Receiver*.

LRMP API

LRMP API provides functions so any received or sent packet behaves as a reliable packet using multicasting.

Socket

Socket element is responsible for creating a socket using LRMP API by communicating with LRMP API element. *Socket element* will encapsulate each data item intended to be sent out to *Sender element.* Each data item converted from *Converter element* is encapsulated by *Socket element. Socket element* also decapsulates each received packet from *Receiver element.*

Converter

Converter element is responsible for converting any data from the client/server which comes from the application interface to UTF8[1] form before passing it to *Socket element.* It also converts decapsulated data from *Socket element* into form that is understood by internal programs on a server/client.

Sender

Sender element is responsible for receiving packets from *Socket element* and sending it to client/server using multicasting.

Receiver

Receiver element is responsible for listening to any received packet from client/receiver and passing it to *Socket element.*

[1]UTF8 is a compact binary form for encoding 16-bit Unicode characters into 8 bits

3.7 Flow Design of WebCOM

This section explains flows in the WebCOM system according to the WebCOM design described in previous section. The WebCOM system flows are divided into four categories based on the objects and actions. The categories are:

- WebCOM Client as sender
- WebCOM Client as receiver
- WebCOM Server as sender
- WebCOM Server as receiver

3.7.1 WebCOM Client as A Sender

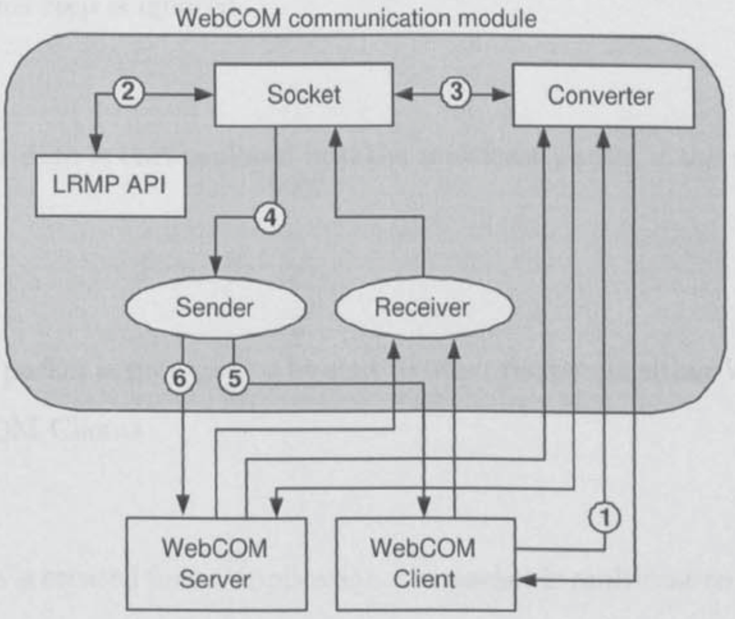


Figure 3.7: Flow of WebCOM Client As A Sender

Figure 3.7 shows the flows of a WebCOM Client that acts as a sender in the WebCOM system. The flows of the WebCOM Client are numbered from 1 to 6. The descriptions of the flows are explained as follow:

- **Step 1:**

  Information used at the beginning of WebCOM Client execution are IP address, port number, and TTL value. These values are converted into appropriate internal data type i.e integer for port number and TTL[2] value, and IP format for IP address. These information are used once only by the WebCOM Client at every initial execution. After the WebCOM Client is executed, the information comes from applications that are connected to the WebCOM Client. This information is converted into UTF8[3] data format for internal used.

- **Step 2:**

  A socket is created by importing the LRMP API classes which create and enhance the reliability of multicast packet. This process is done at the initial stage at the beginning of every WebCOM execution. If the multicast socket has already been created, this step is ignored.

- **Step 3:**

  The UTF8 data is encapsulated into the multicast packet if the multicast socket is created.

- **Step 4:**

  Multicast packet is prepared to be sent to other recipients either WebCOM Server or WebCOM Clients.

- **Step 5:**

  If the data is created for an application, the packet is multicast to other WebCOM clients.

- **Step 6:**

  If the packet is for WebCOM management (such as users and system maintenance), it will multicast the packet to the WebCOM Server.

[2]as mentioned in Section 2.5

[3]one of the data type in programming language

3.7.2 WebCOM Client as A Receiver

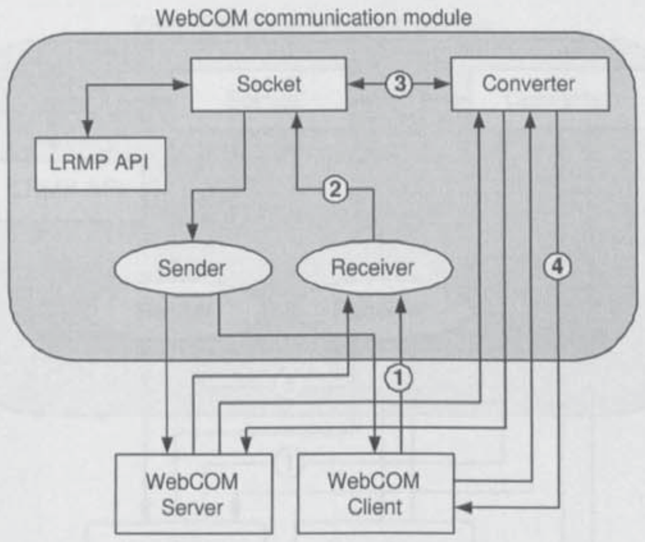


Figure 3.8: Flow of WebCOM Client As A Receiver

Figure 3.8 shows the flows for a WebCOM Client that acts as a receiver in the WebCOM system. The flows of the WebCOM Client are numbered from 1 to 5. The descriptions of the flows are as follows:

- **Step 1:**

  WebCOM Client acts as a daemon for the system and is ready to receive any packet from network. Once it receives a packet from the network, the packet is passed to the "Receiver" component for further processing.

- **Step 2:**

  The received packet is decapsulated into UTF8 data format.

- **Step 3:**

  The UTF8 data format is converted into an internal data type that depends on the type of application used.

- **Step 4:**

  The converted data is passed to the WebCOM Client and is used in the application requiring that data.

3.7.3 WebCOM Server as a Sender

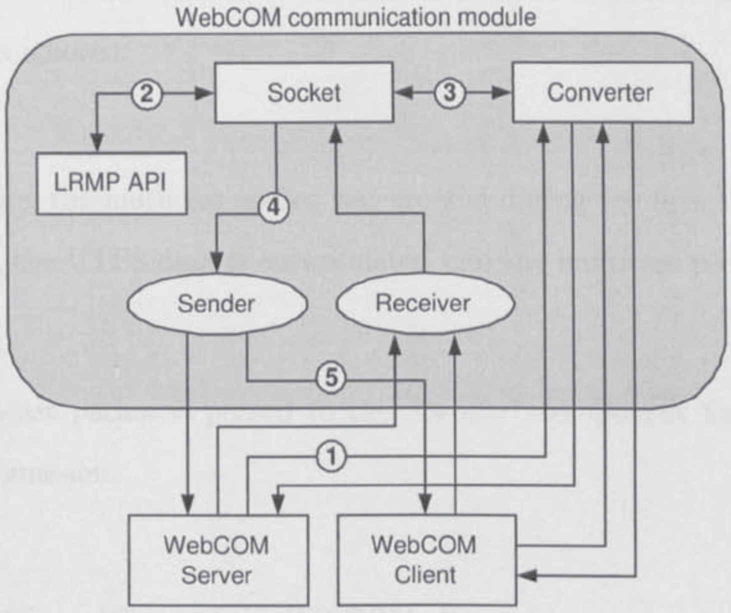


Figure 3.9: Flow of WebCOM Server As A Sender

The WebCOM Server needs to be executed at startup of the WebCOM system, before WebCOM Clients, because it handles all databases for the system. Figure 3.9 shows the flows for a WebCOM Server that acts as a sender. The flows of WebCOM Server are numbered from 1 to 4. The descriptions of the flows are as follows:

- **Step 1:**

  At the beginning of WebCOM Server execution, the information used is IP address, port number, and TTL value. These value are converted into appropriate data type i.e integer for port number and TTL value, and IP format for IP address. After execution of the WebCOM Server, the information from the WebCOM Server is taken from WebCOM databases and the WebCOM system. This information is converted into UTF8 data format in order to encapsulate into a multicast packet.

- **Step 2:**

  The socket is created by importing the LRMP API class to create and enhance the reliability of multicast packet. This process is done at initial stage at the beginning of WebCOM Server execution. If the multicast socket has been created, this step is ignored.

- **Step 3:**
  By assuming the multicast socket was created during the first WebCOM Server execution, the UTF8 data is encapsulated into the multicast packet.

- **Step 4:**
  The multicast packet is passed to the "Sender" component for preparation of packet submission.

- **Step 5:**
  The packet is multicast to the WebCOM clients.

3.7.4 WebCOM Server as A Receiver

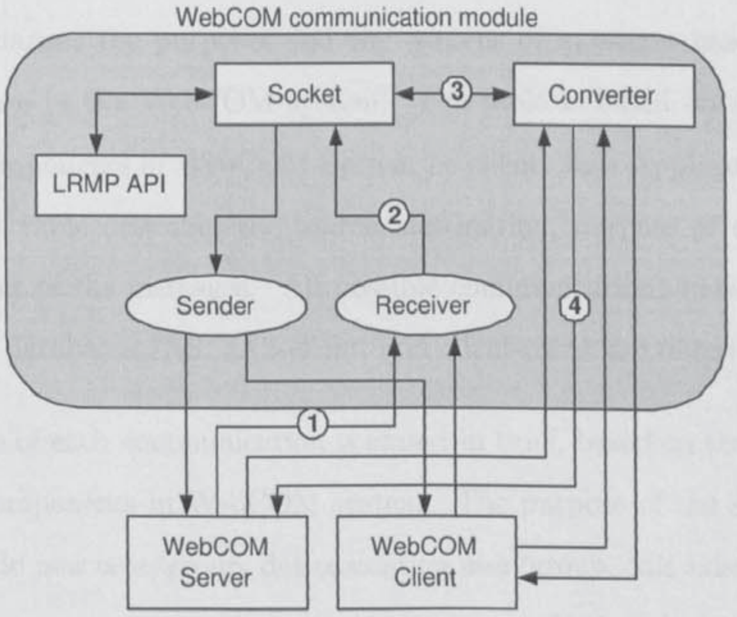


Figure 3.10: Flow of WebCOM Server As A Receiver

Figure 3.10 shows the flows for a WebCOM Server that acts as a sender. The flows of WebCOM Server are numbered from 1 to 5. The descriptions of the flows are explained as follow:

- **Step 1:**

  The WebCOM Server acts as a daemon for the WebCOM system and ready to receive multicast packet from network. Once received, the packet is passed to the "Receiver" component for further processing.

- **Step 2:**

  The received packet is decapsulated into UTF8 data format.

- **Step 3:**

  The UTF8 data format is converted into internal data type.

- **Step 4:**

  The converted data is passed to the WebCOM Server to be processed.

3.7.5 Data flows for WebCOM System Communication

Table 3.3 summarises the purposes and the content of messages based on the state of communication in the WebCOM system. The table is based on the communication between components in WebCOM System i.e client, Java Application Server, and databases. The table describes the source, destination, purpose of the communication, and content of the messages. All possible communications between client-JAS, JAS-databases, databases-JAS, JAS-client, and client-client are described.

The purpose of each communication is stated in brief, based on the possible activities between components in WebCOM system. The purpose of the activities can be summaries as add new user/group, delete existing user/group, edit existing user/group, update existing user/group, and user authentication. Communication via multicasting happened between client-client, client-JAS, and JAS-client. The message contents of this communication is data and control. Internal communication is achieved using internal command with parameters.

Table 3.3: Detail Data Flows of WebCOM System Communication

Source	Destination	Purpose	Content of message
Client	Java Application Server	User & Group Information: Add, Delete Edit/Update Authentication Request Events	Control and related data string Control
Java Application Server	Databases	User & Group Information: Add, Delete Edit/Update Authentication Request Events	Internal command with parameter and data string Internal command with parameter
Databases	Java Application Server	User & Group Information: Response on Add, Delete, Edit, Update, and Authentication Response to Request Events	Internal command with data string(status) Internal command with data
Java Application Server	Client	User & Group Information: Response on Add, Delete, Edit, Update, and Authentication Response to Request Events	Control and related data string
Client	Client	Application functions: Send/receive messages Send/receive drawing	Control and related data string

3.8 Design of Prototype

The objective of the prototype design is to provide a groupware application on the Internet which can be used by any authorised user in a group from anywhere and at anytime. The design is considered in term of the user activities within a group. Unified Modelling Language (UML) is used to describe the design of the prototype system. Two types of UML diagrams are used to describe the design, namely, an activity diagram and a sequence diagram.

3.8.1 The Aim of the Prototype

The aim of the proposed prototype is to provide a generic CSCW application which offers basic tools for collaborative work in the Internet. The tools must support asynchronous and synchronous modes of interaction and geographic diversity as discussed in Chapter 2, Section 2.2.

3.8.2 Existing and Prototype Tools

CSCW applications support a user tasks by providing tools. There are many tools available to support different user's tasks. For example, a drawing tool is used to draw a drawing with a set of options such as colour, width and length of line. However, in this prototype only three tools are considered. The choice of tools is based on selecting tools that operate in asynchronous and synchronous works (explained in Section 2.2). This study concentrates on these three components/tools which are included in the prototype:

- bulletin board - a tool that is used to give information to users about something that is going to happen or that has happened.
- chat - a tool used between users in a group to communicate. This method is used in many CSCW applications.
- shared workspace - a tool that is used to share a single piece of work within a common shared workspace between users in a group.

3.8.3 System Overview

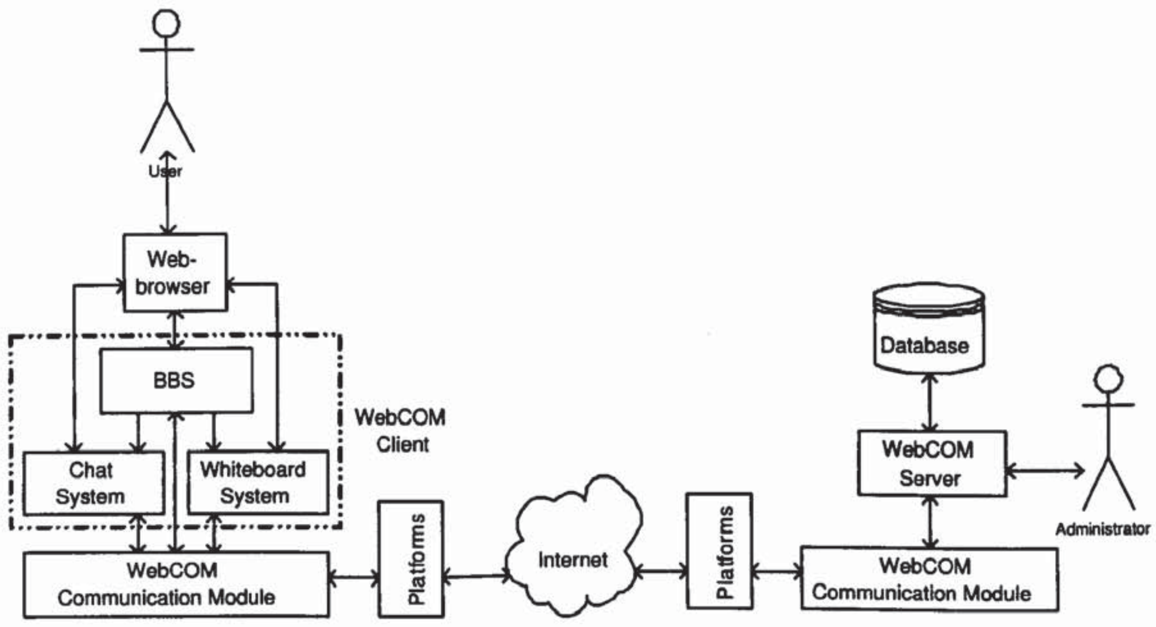


Figure 3.11: Overall View of the WebCOM System

The overall view of the WebCOM System is shown in Figure 4.3. The WebCOM System runs on any platform which is connected to the Internet. Users as actors interact with the WebCOM System using a Web browser as a user interface. While the administrator interacts with WebCOM Server directly using local console to manage the database. The WebCOM Server and database reside on different places from WebCOM Client and can be accessed through the Internet.

Figure 3.12 shows the protocol stack used for the prototype system. The prototype system runs on top of WebCOM, which is responsible for receiving and sending data/control using multicasting. TCP/IP and UDP/IP are used as usual to support WebCOM communication with other parties in the Internet.

3.8.4 Assumption of Prototype Design

As mentioned in Chapter 1, one of the advantages of the WebCOM System is *accessibility* in which participants are able to work even when there is no available server. How-

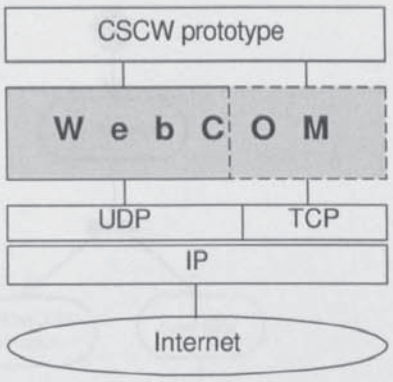


Figure 3.12: Protocol Stacks Used by Prototype

ever, for design purposes, the work in this thesis had made an assumption below:

- participants are connected to a Web server.
- Web server is operational.

3.8.5 Activity Diagrams

The activity diagram in this thesis is used to explain the flow of activity from beginning to end. Figure 3.13 shows the activity diagram for the prototype. At the initial stage, a user accesses a Web page (containing information and a link into the WebCOM System) from a Web server by entering the address URL (Uniform Resource Locator) of the prototype. From the link, the WebCOM Bulletin Board System (BBS) appears on a Web page in which all functions are disabled except login and user register functions. If the user is a registered member of the WebCOM System, he/she can immediately logon to the system. Otherwise, he/she needs to register on the system. After a successful logon to the system, all functions are enabled and available for use except login and new user functions.

An interface for the WebCOM BBS shows a set of empty columns. In this page, six options are available (see Figure I.2):

- Register - register a new user into the WebCOM System.

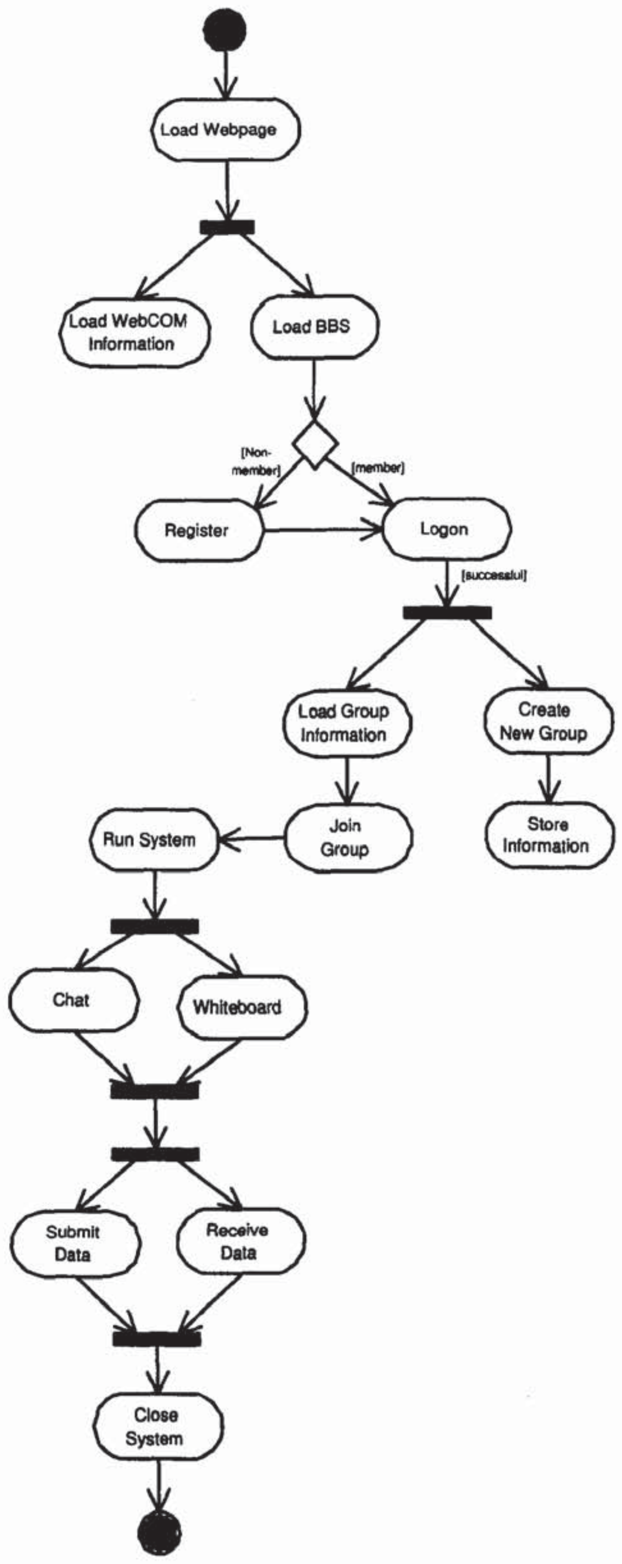


Figure 3.13: Prototype Design: Activity Diagrams

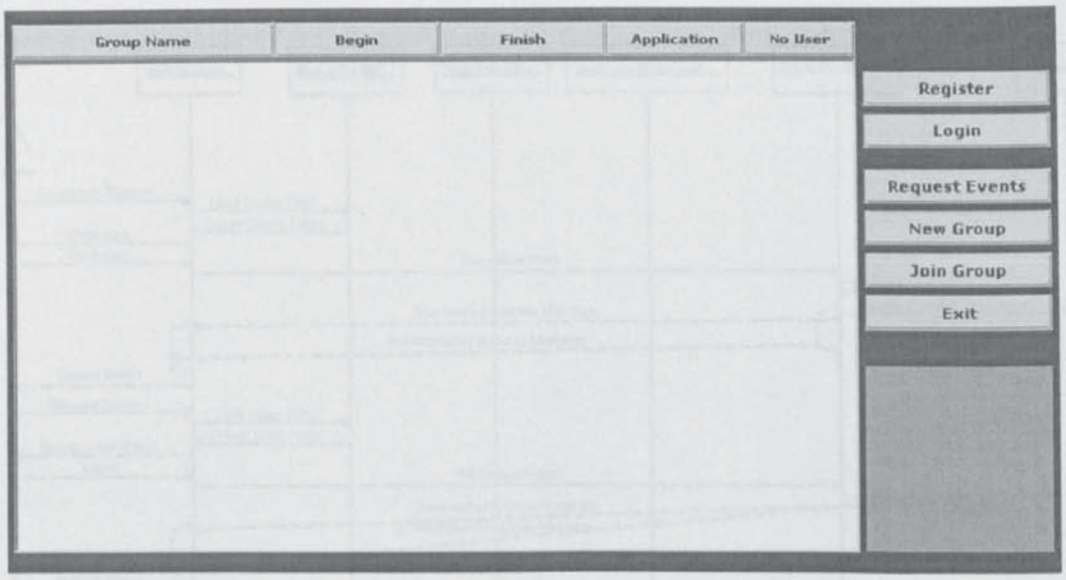


Figure 3.14: Bulletin Board System with Disable Functions

- Login - logon into the WebCOM System to use the WebCOM System.
- Request Group Information - load available information on group discussion in the WebCOM System.
- Create New Group - create a new group, with some information on it. Information on the new group will be stored in the database.
- Join Group - join a selected group which is available in the WebCOM System.
- Exit - exit from the WebCOM System.

Once participants request information on groups available in the WebCOM System, all information will appear on the WebCOM BBS interface. The information is used to connect participants to either the WebCOM Chat system or WebCOM Whiteboard system, depending on the selection. Each application is able to send and receive data from the same application in the same group. Any participant can leave the system by closing the active windows.

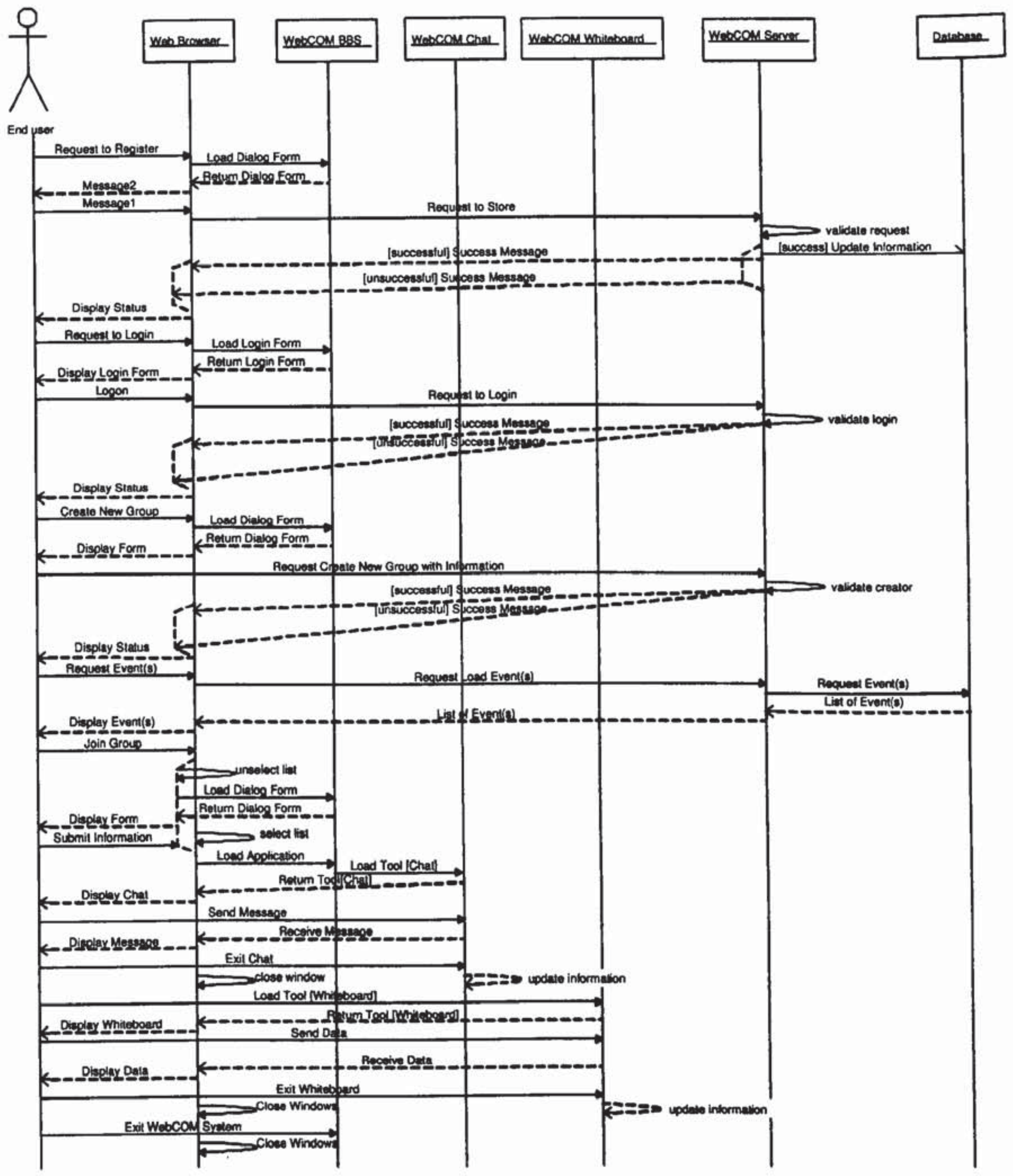


Figure 3.15: Prototype Design: Sequence Diagrams

3.8.6 Sequence Diagrams

As the name suggests, a sequence diagram describes the design in term of an activity sequence, which comes first and which comes after. Figure 3.15 shows the sequence diagram for the WebCOM System. Initially, a user requests Web pages that contain information on the WebCOM System by entering the URL address for the WebCOM System at a Web browser. The Web browser displays the requested page which contains information on WebCOM System and a link to WebCOM System. The first page of WebCOM System is WebCOM BBS with enable functionality except functions of Register and Login. If the user is non-member for the WebCOM System, they need to register on the system by requesting the registration form. On completing the form, the information will be stored in the database, which is updated with the user information.

If the user is a registered member for the WebCOM System, he/she can directly logon into the system. When a user logons the system, he/she will be validated by the WebCOM Server by checking the information inside a database. If logon information does not match with the information in the database, a failure message is returned to the user by displaying it on the WebCOM BBS. If the logon process is successful (where data entered for logon matches information in the database), a successful message is returned to the user by displaying it on the WebCOM BBS.

Creating a new group is straight forward. The user asks to create a new group and the WebCOM BBS will display a form. When the new group information is submitted, the information details of that new group is updated in the database.

The user can check the availability of a group in the system by sending a request from the WebCOM BBS to the WebCOM Server. The WebCOM Server then loads the information from a database and send it to the WebCOM BBS. If the information is not available, a failure message (error message) is returned to the WebCOM BBS.

On the WebCOM BBS, information for all groups available in the system are displayed. A user can join any group by selecting the group on the list of WebCOM BBS or manually entering the information on a specific known group. Each group will be associated with one of the applications, either WebCOM Chat system or WebCOM Whiteboard system. The WebCOM Chat system and WebCOM Whiteboard system work using the same method for sending and receiving data/message to/from other participants in that group. The difference between these applications is in internal data/message manipulation, which is hidden from the user.

Each user can leave the group by closing the active window running an application associated with a particular group. A leaving message is then submitted to other participants in same group.

Chapter 4

Implementation of the WebCOM System

4.1 Introduction

This chapter describes how WebCOM System is implemented as a communication framework for the CSCW application prototype. The WebCOM System is implemented by running a prototype of CSCW applications on WebCOM, which acts as a communication framework for the prototype. The WebCOM System is constructed from a WebCOM Communication Module, WebCOM Client, and WebCOM Server (as shown previously in Figure 3.2). The Communication Module will be present in both client and server. The prototype CSCW application is constructed using three main components/tools: chat, whiteboard, and bulletin board.

The WebCOM System and each prototype component was built using the Java language and Java API to take advantage of the platform independence of Java. The WebCOM System has been implemented using Java applets in the Web browser. In this way, all Java-enabled browsers can use the system at any place, at any time and on any platform.

4.2 WebCOM Communication Framework

This section describes the complete WebCOM Communication Framework which consists of three components as shown in Figure 4.3: WebCOM Client, WebCOM Server, and WebCOM Communication Module. All components are implemented using the Java language and Java APIs.

The WebCOM Server will initiate itself to the interconnected users before any users can use the WebCOM system. CSCW tools integrated with the WebCOM Client can be used once the WebCOM Client has joined the interconnected group. The communication between WebCOM Client and WebCOM Client or between WebCOM Server and WebCOM Client is via WebCOM Communication Module and uses multicasting. Figure 4.1 shows the summaries of components present, state and message passing involved in WebCOM System using UML sequence diagram.

4.2.1 WebCOM Client

The WebCOM Client provides a framework for CSCW applications to be executed in the WebCOM system. The architecture of a WebCOM Client was shown previously in Figure 3.3, which shows three main functions in a WebCOM Client:

- CSCW tools integration

  CSCW tools can run on top of the WebCOM Client. With this, any function of CSCW tools can be integrated with WebCOM Client. All underlaying functions of CSCW tools is handled by the WebCOM Client. These include updating internal events, managing internal/external events, and listening internal events.

- Group joining

  WebCOM Client provides the end-user with a function to join a specific group in a network. As illustrated in the next paragraph, WebCOM Client uses a multicast socket to join a group by sending an IP address and a port number of that group, which has been acquired from WebCOM Server. There are three ways to acquire this information. Firstly, the information is stored as parameters in a web

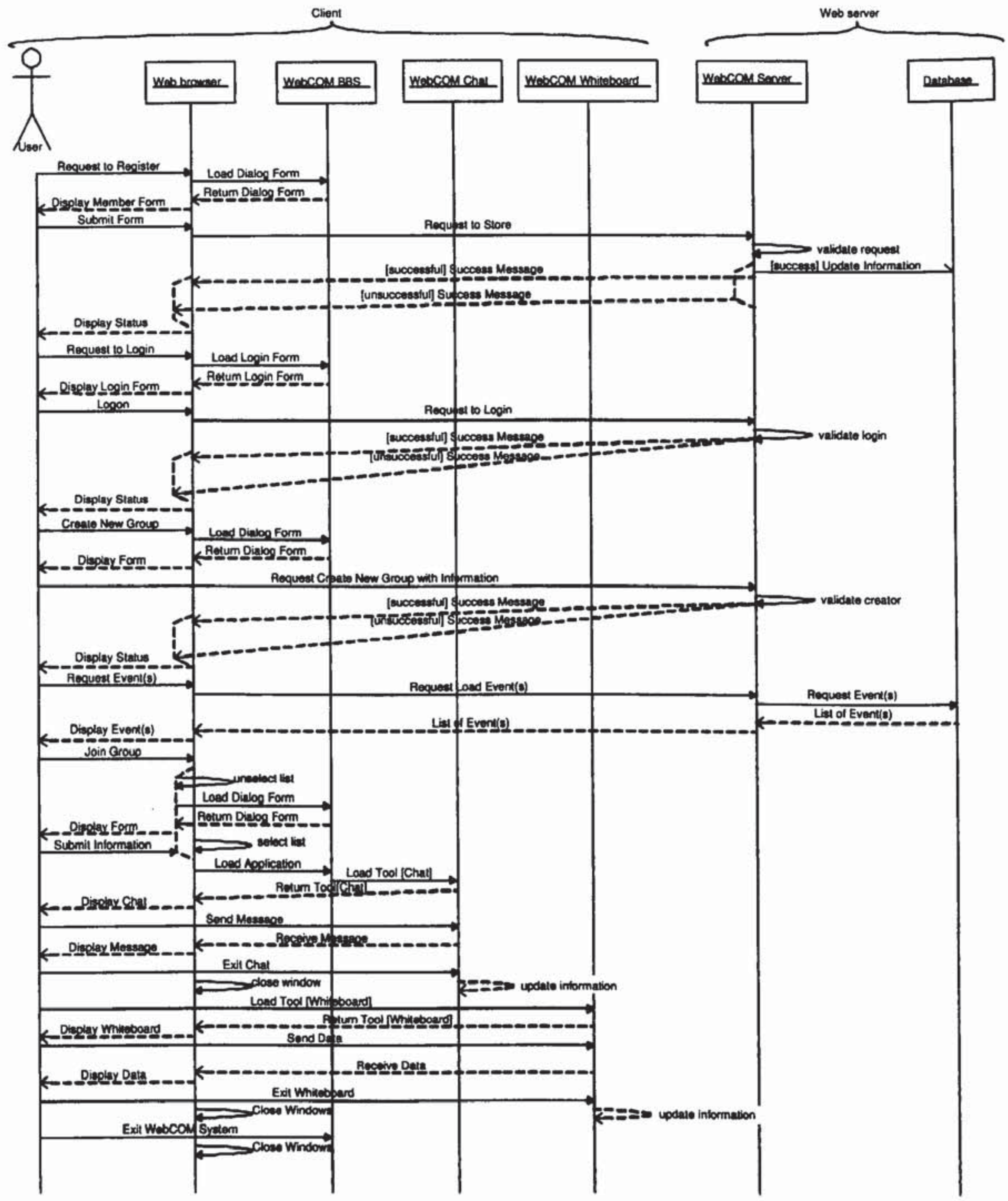


Figure 4.1: Sequence Diagram of WebCOM System: Components, States and Operations

page downloaded from the server. The information is used by WebCOM Client named Bulletin Board System(BBS). Secondly, the information is stored together with a link of event that appears on the WebCOM Client BBS downloaded from WebCOM databases at the server. Thirdly, the information is acquired from users by keying the information into a dialog box shown in Figure 4.2.

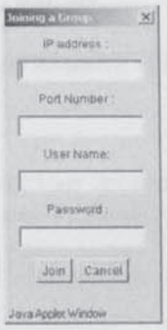


Figure 4.2: Dialog Box for Joining a Group

The group joining function is established when a multicast socket is created by a WebCOM client. The multicast socket is created by passing an IP address, a port number, and a Time To Live(TTL) value to the WebCOM communication module. Below is the pseudo-code for the group joining function:

```
/*
Purpose: pass the IP address, port, and TTL value to
join a specific WebCOM Client.
Result: the client will join an interconnected group
specified by the information($ipAddress$ and
$port$ value) and a WebCOM Client associated with the
group will be executed on their screen.
*/

WebCOM $CSCWtool$ = new WebCOM($ipAddress$,$port$,$ttl$);
```

- Group leaving

  WebCOM Client provides the end-user with a function to leave a group they are currently joined to by sending a 'leaving' message to the group. Below is the pseudo-code for the group leaving function:

```
/*
Purpose: send a leaving message to the interconnected users
(or group) to leave the group.

Result: the user will terminate from the joined group.
*/

/*
Purpose: convert the leaving message into UTF8 data
format and assign it into variable $data$.

Result: variable $data$ consists of message in UTF8 data
format.
*/

byte data[] = $messageOfLeaving$.getBytes("UTF8");

/*
Purpose: prepare a datagram packet with required information
 associated with variables :$data$,$ipAddress$ and $port$.

Result: a datagram packet associated with variable
$datagramPacket$ is prepared with required
information and is ready to be sent out.
*/

DatagramPacket $datagramPacket$ = new DatagramPacket
    ($data$,$data$.length, $ipAddress$, $port$);

/*
Purpose: send the datagram packet to interconnected group
using variable $datagramPacket$.

Result: $datagramPacket$ is multicast to others in
interconnected group via established multicast socket, $socket$.
*/

$socket$.send($datagramPacket$);

/*
Purpose: close the socket, $socket$ after sending the data to
the interconnected users/group.

Result: the socket, $socket$ will be closed.
*/

$socket$.close();
```

4.2.2 WebCOM Server

The WebCOM Server software is located on a Web server. Its responsible for activities that involve database interactions. There are four main functions in WebCOM Server:

- Server initialisation

  WebCOM Server provides an automatic function to publish itself to the WebCOM system. As illustrated in the next paragraph, WebCOM Server uses a multicast socket providing an IP address, a port number, and TTL value to initiate itself to the interconnected users, so that other users can be connected. This information is assigned by WebCOM system administrator. Below is the pseudo-code for the WebCOM Server initiation function:

```
/*
Purpose: obtain IP address, port number and TTL value from
database located in server and created by WebCOM system
administrator (WebCOM system administrator needs to provide
this information in the database before WebCOM Server can
start).  The information is used to create a multicast socket
used to initiate the WebCOM Server.
Result: WebCOM server will be executed on the server side.*/

/*
Purpose: open database file, $file_name$ that was created by
WebCOM Administrator on a server.
Result: database file is open and ready to be used with
associated variable, $database$.*/

FileReader $database$= new FileReader("$file_name$");

/*
Purpose: locate variable $database$ into a buffer variable,
$initialState$ in order to manipulate the data.
Result: variable $initialState$ consists of information in
the database.*/

BufferedReader $initialState$ = new BufferedReader($database$);

/*
Purpose: manipulate the information in the buffer to get IP
address and port number and convert them into appropriate data
type to be used to create the socket.
Result: variables $port$ holds a port number and $IPaddress$
holds a IP address for the WebCOM server.*/

integer $port$ = Integer.parseInt($portValue$);
InetAddress $IPaddress$ = InetAddress.getByName($ipAddressValue$);

/*
Purpose: call the create socket function at WebCOM
Communication Module using these information($port$ and
$IPaddress$) to initialize the WebCOM server.
Result: WebCOM server is initiated and executed on the server.*/

createSocket();
```

- Server termination

  WebCOM Server provides a function to leave as a server for WebCOM system by sending a 'leaving' message to interconnected users. WebCOM clients currently connected to the WebCOM system will be disconnected from the WebCOM server and all processes that involved databases will not function. However, other functions of WebCOM Client work as normal although there is no connection to the WebCOM server. In other words, WebCOM Clients are able to communicate independently of the WebCOM Server once they have suitable data and software. Thus the group can operate with the server no longer in operation. Below is the pseudo-code for this function:

```
/*
Purpose: send the leaving message to the interconnected users
to terminate the server service for WebCOM System.
Result: the WebCOM server will be terminated and the group
can operate with the server no longer operation.*/

/*
Purpose: prepare a leaving message which is assigned as
$messageOfLeaving$ and convert it into UTF8 data format
which is assigned as $data$.
Result: variable $data$ consists of message of leaving in
UTF8 data format is prepared.*/

byte $data$[] = $messageOfLeaving$.getBytes("UTF8");

/*
Purpose: prepare the information in $data$ into datagram
packet, $dp$ to be sent to other clients.
Result: datagram packet, $dp$ is prepared and ready to send
out to other clients. */

DatagramPacket $dp$=new DatagramPacket($data$,$data$.length,
$IPadress$,$port$);

$socket$.send(dp);

/*
Purpose: terminate the WebCOM Server and close the established
socket.
Result: the WebCOM server is terminated from interconnected
group and the established socket is closed. */

$receiver$.terminate();
$socket$.close();
```

- Incoming and outgoing packet Management

  WebCOM Server provides a function of managing any incoming and outgoing packets at the server. WebCOM Server acts as a daemon that keeps listening to incoming packet. All incoming packets are filtered by WebCOM Server to identify the type of a packet as either data packet, control packet, or both data and control packet. A data packet must comes with its control, however a control packet need not have data. The control packet defines what tasks should be processed by WebCOM Server. If an incoming packet is undefined, it will be discarded and an error message is multicast to the client.

  Data for an outgoing packet is prepared by WebCOM Server from databases on a same Web server. The data and associated control created by the WebCOM Server is passed to the WebCOM Communication Module for further processing. Below is the pseudo-code for this function:

```
/*
Purpose: listen for incoming packet (request from client) using
established multicast socket, $socket$.
Result: the WebCOM server will keep listening for incoming packets
from the established socket connection, $socket$.  If any
incoming packet occurs, the datagram packet will be assigned
into variable $dp$. */

DatagramPacket $dp$ = $socket$.receive();

/*
Purpose: read datagram packet, $dp$ and transfer it into a
variable, $message$.  The $message$ is transferred to string
buffer, $messageBuffer$ in order to manipulate it.
Result: a string buffer, $messageBuffer$ consists of string
in UTF8 data format is prepared and ready to be manipulated. */

String $message$ = new String($dp$.getData(), "UTF8");
StringBuffer $messageBuffer$ = new StringBuffer($message$);

/*
Purpose: pick the control word within a string buffer,
$messageBuffer$.
Result: a variable, $strCheck$ consist of control word is
created. */

String $strCheck$ = $messageBuffer$.substring(0,20);
```

```
/*
Comments: the packet's information in buffer is
implemented based on the control word, which is a request for
events list, creating new user or new group.  The information
then is stored in a database or fetched from a database on the
same server, depending on the control word. */

/*
Purpose: check the control word, $strCheck$ in order to define
what action should be performed.
Result: the control word, $strCheck$ direct the action
should be performed. */

case ($requestStatus$) of
/*
Purpose: control word is requesting for list of events.\\
Result: action to get a list of events from database on
server will be performed. */

    listRequest:

Purpose: get the information from database & assign
it as $messageList$.  Then convert it into UTF8 data type,
$dataList$.
Result: $dataList$ consists of a list of events is created
and prepared to sent out. */

$dataList$ = $messageList$.getBytes("UTF8");

/*
Purpose: prepare a datagram packet for $dataList$ and assign
it as $dpList$.
Result: $dpList$ contains all required data, $dataList$,
$IPaddress$, and $port$ number.*/

DatagramPacket $dpList$ = new DatagramPacket($dataList$,
$dataList$.length, $IPaddress$, $port$);

/*
Purpose: send the information to the clients
Result: the packet is multicast to connected clients*/

$socket$.send($dpList$);

/*
Purpose: control word is requesting creation of a new user.
Result: action to get store for a new user into database on
server will be performed. */

    newUserRequest:

/*
Purpose: open a database on the server and assign it to a
variable $database$.
Result: $database$ variable holds the database. */

FileWriter $database$= new FileWriter($databaseName$,true);
```

```
/*
Purpose: assign the $database$ to a buffer $userData$ in order
to manipulate the information inside the database.
Result: a buffer $userData$ is created. */

BufferedWriter $userData$ = new BufferedWriter($database$);

/*
Purpose: write the new user record from buffer packet
$messageBuffer$ into the database, $userData$.
Result: user information is stored in the database. */

$userData$.write($messageBuffer$);

/*
Purpose: close the database.
Result: the database is closed. */

$userData$.close();

/*
Purpose: control word is requesting creation of a new group.
Result: action to store a new group in database on
server will be performed. */

    newGroupRequest:
/*
Purpose: open a database on the server and assign it to a
variable $database$.
Result: $database$ variable holds the database. */

FileWriter $database$= new FileWriter($databaseName$,true);

/*
Purpose: assign the $database$ to a buffer $groupData$ in
order to manipulate the information inside the database.
Result: a buffer $groupData$ is created. */

BufferedWriter $groupData$ = new BufferedWriter($database$);

/*
Purpose: write the new user record from buffer packet
$messageBuffer$ into the database, $groupData$.
Result: user information is stored in the database. */

$groupData$.write($messageBuffer$);

/*
Purpose: close the database.
Result: the database is closed. */

$groupData$.close();


/*
Purpose: control word is requesting login to the WebCOM
System.
Result: user's status either valid or invalid to enter
the system. */
```

```
    loginRequest:
/*
Purpose: compare the login name received
and login name in the database to authenticate the users.
Result: status of users either valid or invalid to use the
WebCOM System using this information. */

boolean $checkLogin$ = $usernameUser$.equals($loginInDatabase$);

Purpose: compare the password received
and password in the database to authenticate the users.
Result: status of users either valid or invalid to use the
WebCOM System. */

boolean $checkPassword$ =
$passwordUser$.equals($passwordInDatabase$);

Purpose: convert $message$ that contains information on user
status into UTF8 data format.
Result: variable $data$ holds the information on user's status
in UTF8 data format. */

$data$ = $message$.getBytes("UTF8");

Purpose: prepare the datagram packet ($datagramPacket$) to
carry the status result($data$).
Result: datagram packet, ($datagramPacket$) is created. */

$datagramPacket$ = new
DatagramPacket($data$,$data$.length,$ipAddress$,$port$);

Purpose: send the $datagramPacket$ to other interconnected
users.
Result: datagram packet, $datagramPacket$ is multicast to
other users. */
$socket$.send($datagramPacket$);
```

- Database[1] Management.

  The WebCOM Server provides a number of functions to maintain WebCOM System databases that reside on the same server. These tasks are executed depending on a control packet received by the WebCOM Server. The functions are:

  - add new use or group:

    This function can be accessed by users from WebCOM Client BBS.

  - delete existing user or group:

    This function can be accessed by administrator only at server side.

[1]WebCOM Server currently uses text files as databases for the system. Appendix I Section I.10.2 describes the databases.

  - check authorisation of users:
    This function is run by the WebCOM Server automatically when users login to the system.
  - response to any request from clients:
    This function is responds to any request based on the control packet received from the participants.

4.2.3 WebCOM Communication Module

WebCOM Communication Module is used by WebCOM Clients and a WebCOM Server to enable communication between group participants in a WebCOM System. The architecture of WebCOM Communication Module was shown previously in Figure 3.6. There are five main functions in WebCOM Communication Module:

- Socket creation
  Communication in WebCOM system is established by creating a multicast socket to the interconnected users. A multicast socket is created by using an LRMP API that enables full-duplex communication on multicasting. A multicast socket is prepared by passing establishment parameters to the LRMP API. The LRMP API is a collection of Java classes to support reliable multicasting communication. In order to create create a multicast socket, IP address, port number, and Time-To-Live (TTL)[2] values are required. Below is the pseudo-code for the socket creation:

[2]as mentioned in Section 2.5

```
/*
Purpose: initialize an IP address and port number value to be
used in multicast socket creation.
Result: valid IP address and port number which can be used in
the programming. */

InetAddress $IPaddress$=InetAddress.getByName($IPaddressValue$);
integer $port$ =Integer.parseInt($portValue$);

/*
Purpose: create LRMP transport profile in order to create
reliable multicast socket.
Result: reliable multicast socket is created. */

TransportProfile $tp$ = null;

/*
Purpose: create a new LRMP transport profile, $LRMPtp$ using
IP Address and port number.
Result: LRMP transport profile, $LRMPtp$ is created. */

LRMPTransportProfile $LRMPtp$ = (LRMPTransportProfile)
     new LRMPTransportProfile($IPaddress$,$port$);

/*
Purpose: set the LRMP transport profile, $LRMPtp$ with TTL
value.
Result: $LRMPtp$ is created using a new TTL value.  This
value will indicate how long the packet will appear on the
network. */

$LRMPtp$.setTTL((byte) Integer.parseInt($ttlValue$));

/*
Purpose: set the LRMP transport profile, $LRMPtp$ with either
support information/packet ordering or not ordering.  If true,
means support ordering otherwise not support ordering.
Result: LRMP transport profile, $LRMPtp$ with support
ordering is created. */

$LRMPtp$.setOrdered(true);

Purpose: set the RMP packet socket, $socket$ established to
use the transport profile with full duplex.
Result: RMP packet socket wth full duplex is initilized. */
RMPacketSocket $socket$ =
$tp$.createRMPacketSocket(TransportProfile.$SEND\_RECEIVE$);

/*
Purpose: create a new receiver thread, $receiver$ to act as a
receiver that keep listening.
Result: a new receiver thread, $receiver$ is created. */
ReceiverThread $receiver$ = new ReceiverThread();

/*
Purpose: start the reliable multicast socket as receiver.
Result: reliable multicast socket is started. */
receiver.start();
```

- Packet conversion

  Each multicast packet is being algorithm into "UTF8" format before encapsulated into a multicast outgoing packet. For an incoming packet, the multicast packet is converted into internal data format after being algorithm from "UTF8". The Java function used for the packet conversion is:

  ```
  String message = new String(dp_receive.getData(), "UTF8").
  ```

- Send Packet

  After a multicast packet is prepared, it will be multicast in a same group by a user acting as a sender to other users as recipients using an established multicast socket. The recipient is either WebCOM Clients or WebCOM Server. Below is the pseudo-code for the send packet:

  ```
  /*
  Purpose: prepare the information needed to be submitted.
  Result: the information is prepared to be submitted. */

  String $message$ = "$TEXT$";

  /*
  Purpose: convert the information, $message$ into UTF8 data
  format, $data$.
  Result: the information is converted into UTF8 data format,
  $data$. */

  byte $data$[] = $message$.getBytes("UTF8");

  /*
  Purpose: create a new datagram packet, $dp$ and insert the
  information, $data$ into the datagram packet plus related
  information i.e. IP Address and port number.
  Result: datagram packet, $dp$ is created. */

  DatagramPacket $dp$ = new
  DatagramPacket($data$, $data$.length, $ipAddress$, $port$);

  /*
  Purpose: send the datagram packet, $dp$.
  Result: the datagram packet with information is ready to be
  sent out. */

  socket.send(dp);
  ```

- Receive Packet

  The WebCOM Client and WebCOM Server used the "Data Receiving" component to keep listening on the network. They are acting like a daemon to detect whether any incoming packet occurs. If an incoming packet occurs, this component passes the packet to the "Converter" component in the communication module to convert it into internal data type. Below is the pseudo-code for the receive packet function:

  ```
  /*
  Purpose: listen to any incoming packet from established
  $socket$ and assigned it as datagram packet, $dp$.
  Result: incoming packet is received and assigned as $dp$.

  DatagramPacket $dp$ = $socket$.receive();

  /*
  Purpose: convert the information inside packet, $dp$
  into UTF8 data format, $message$.
  Result: the information is converted into UTF8 data format,
  $message$. */

  String $message$ = new String($dp$.getData(), "UTF8");

  /*
  Purpose: create a new string buffer, $msgBuffer$ to hold
  incoming $message$.
  Result: $msgBuffer$ is created to contain incoming information
  that is converted into string to be used internally. */

  StringBuffer $msgBuffer$ = new StringBuffer(message);
  ```

- Socket termination

  Communication in the WebCOM system is terminated by sending a 'leaving' message to the network. The Java function used for this purpose is *socket.close()*.

4.3 WebCOM CSCW Tools

This section describes the implementation of CSCW tools in WebCOM System: WebCOM Bulletin Board System(BBS), WebCOM Chat, and WebCOM Whiteboard. All operations concept for WebCOM Client and WebCOM Communication Module described in previous sections is applied on WebCOM BBS, WebCOM Chat, and Web-

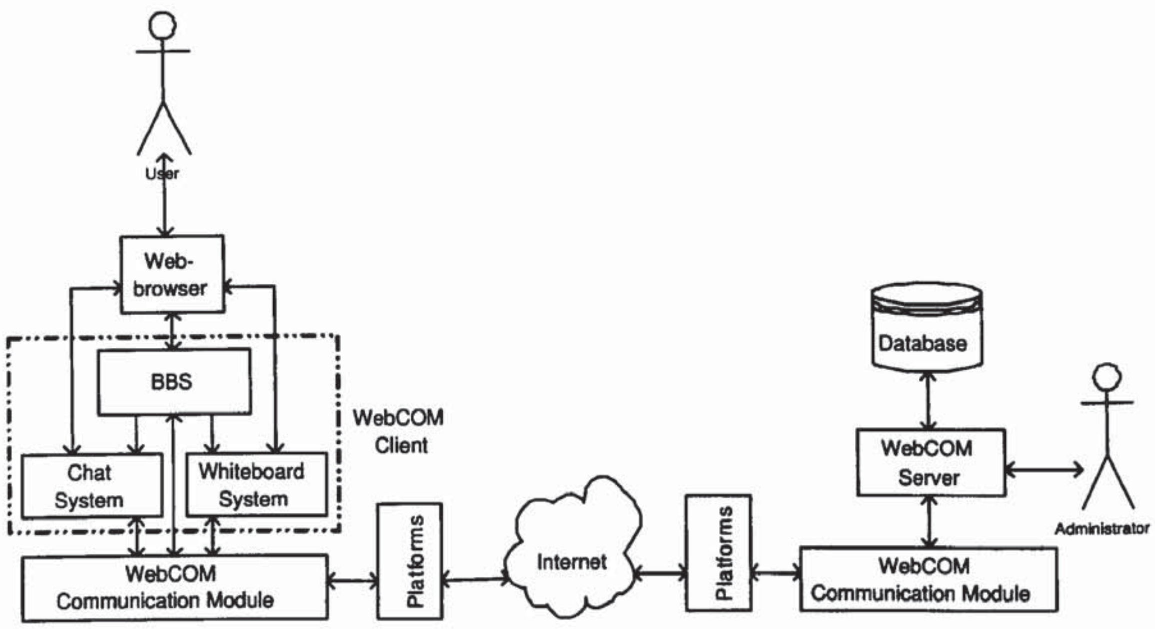


Figure 4.3: Overall View of the WebCOM System

COM Whiteboard. The operations include create new user or group, login, retrieving activity, joining group, exit from system, sending, and receiving information.

4.3.1 WebCOM BBS

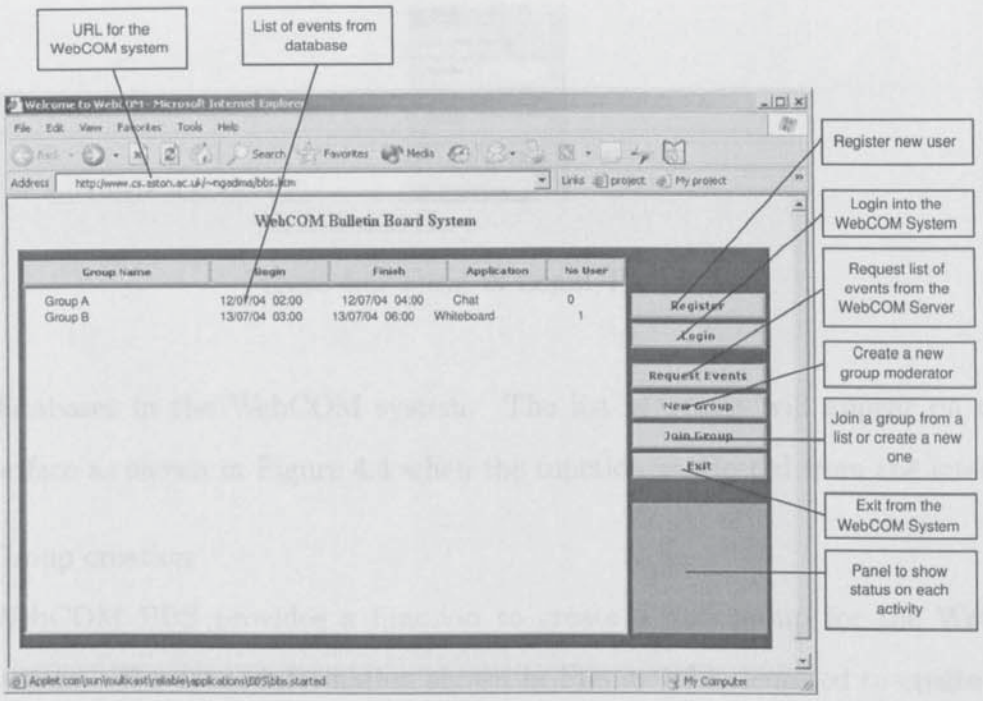
WebCOM BBS contains six functions to enable a user to interact with other participants in a group. The screen layout of WebCOM BBS is shown in Figure 4.4. The functions provides by WebCOM BBS are:

- User registration

  WebCOM BBS provides a function to create a new user as a participant in the WebCOM system by selecting the first top-right button on Figure 4.4. A set of information as stated in Appendix I (Section I.10.2) is required to register as a new user. An example form used to register a new user is shown in Figure 4.5. A user is permitted to register into the WebCOM system only once.

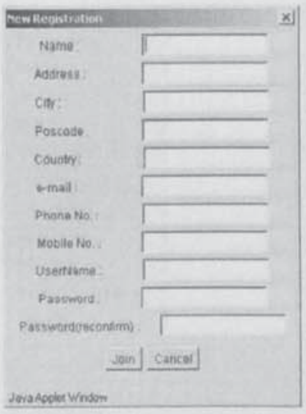
- System login

  WebCOM BBS provides login and password facilities to authorise a valid user


URL for the WebCOM system
List of events from database
Welcome to WebCOM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.cs.aston.ac.uk/~ngadma/bbs.htm
WebCOM Bulletin Board System
Register
Login
Request Events
New Group
Join Group
Exit
Register new user
Login into the WebCOM System
Request list of events from the WebCOM Server
Create a new group moderator
Join a group from a list or create a new one
Exit from the WebCOM System
Panel to show status on each activity
Applet com/sun/multicast/reliable/applications/BBS/bbs started
My Computer

Group Name	Begin	Finish	Application	No User
Group A	12/07/04 02:00	12/07/04 04:00	Chat	0
Group B	13/07/04 03:00	13/07/04 06:00	Whiteboard	1

Figure 4.4: WebCOM BBS Interface



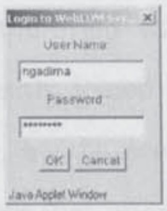
Name :	
Address :	
City :	
Poscode :	
Country :	
e-mail :	
Phone No. :	
Mobile No. :	
UserName :	
Password :	
Password(reconfirm) :	

Figure 4.5: Form of User Registration

access to the WebCOM system. Only a registered user is permitted to login and use the WebCOM system. Non-members are required to register first to the WebCOM system. The form used for authorisation process is shown in Figure 4.6.

- Activity retrieving

  WebCOM BBS provides a function to retrieve a list of events stored in the



Login to WebCOM

User Name

ngadima

Password

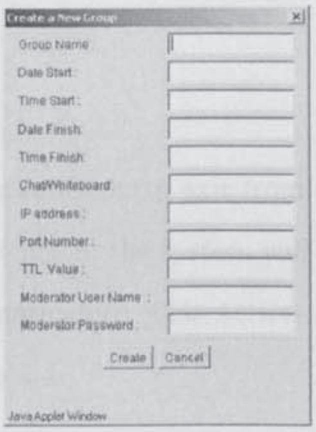
****

Figure 4.6: Form of Login/Password

databases in the WebCOM system. The list of events will appear on the interface as shown in Figure 4.4 when the function is selected from the interface.

- Group creation

  WebCOM BBS provides a function to create a new group for the WebCOM system. The set of information shown in Figure 4.7 is required to create a new group in the WebCOM system. The information will be stored in a database kept on the server.



Create a New Group

Group Name	
Date Start :	
Time Start :	
Date Finish	
Time Finish	
Chat/Whiteboard	
IP address :	
Port Number :	
TTL Value :	
Moderator User Name :	
Moderator Password :	

Figure 4.7: Form of Group Registration

- Group joining

  WebCOM BBS provides a function to join either a group existing in the system or a self-created group (users can create their own group). The existing group in the WebCOM system can be found in a list of events as shown in Figure 4.4. A user can join any group available in the system by selecting one of the events from the list. When the user selects an event from the list, an application (a CSCW tool) associated to the event will be automatically executed on the client side and the user is directly joined to the group for that event. If the group is self-created by a user, he/she needs to provide the set of information shown in Figure 4.8. Other users that intend to join this group needs to provide the same information for that group. Once valid information is provided, the shared application is executed. The shared applications will be WebCOM Chat or WebCOM Whiteboard.

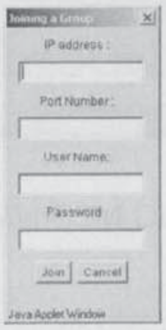


Figure 4.8: Form of Joining a Group

- System exit

  WebCOM BBS provides a function to exit from the current application. When a user exit from that application, the system will terminate them from the group of that application automatically. This action is done by pressing the "Exit" button as shown is Figure 4.4.

4.3.2 WebCOM Chat

The screen layout and operation flows of WebCOM Chat is shown in Figure 4.9 and 4.10 respectively. The WebCOM Chat contains four functions to enable users interact each other in a group. These functions are:

- Input entering

  WebCOM Chat provides a message field at the bottom, as shown in Figure 4.9, for entering user's messages in string form. These messages are forwarded to the WebCOM Communication Module to be multicast to other recipients.

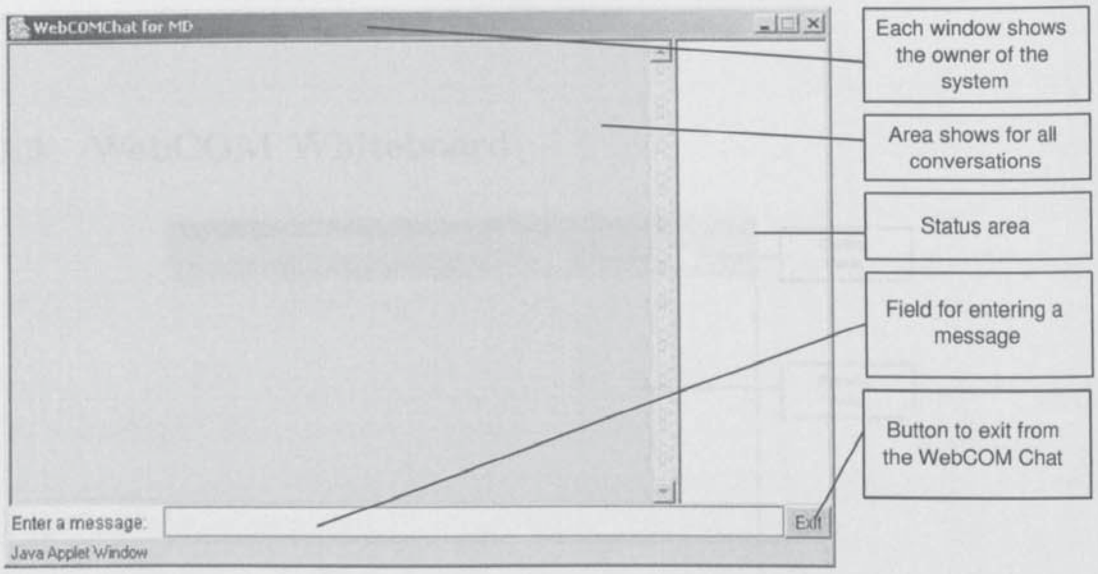

WebCOMChat for MD
Enter a message:
Exit
Java Applet Window
Each window shows the owner of the system
Area shows for all conversations
Status area
Field for entering a message
Button to exit from the WebCOM Chat

Figure 4.9: WebCOM Chat Interface

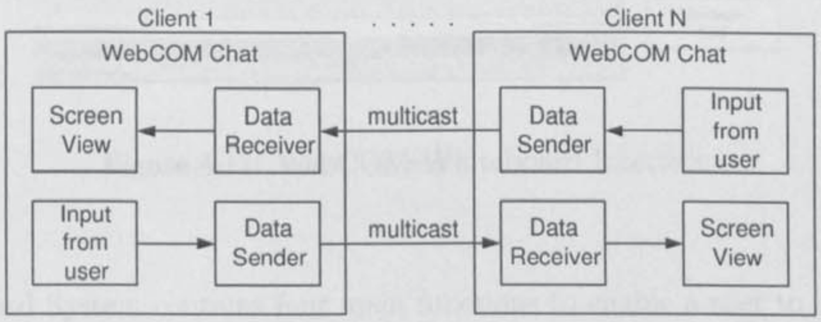


Figure 4.10: Operation Flow of WebCOM Chat

- Data sending

  Each message entered will be sent to a network after pressing "Enter" key.

- Data receiving

  Each incoming WebCOM data will be appeared on the conversation area as shown in Figure 4.9.

- Group leaving

  WebCOM Chat provides a function to exit from current application. When a user exit from that application, the system will terminate them from the group of that application automatically. This action is done by pressing the "Exit" button as shown is Figure 4.9.

4.3.3 WebCOM Whiteboard

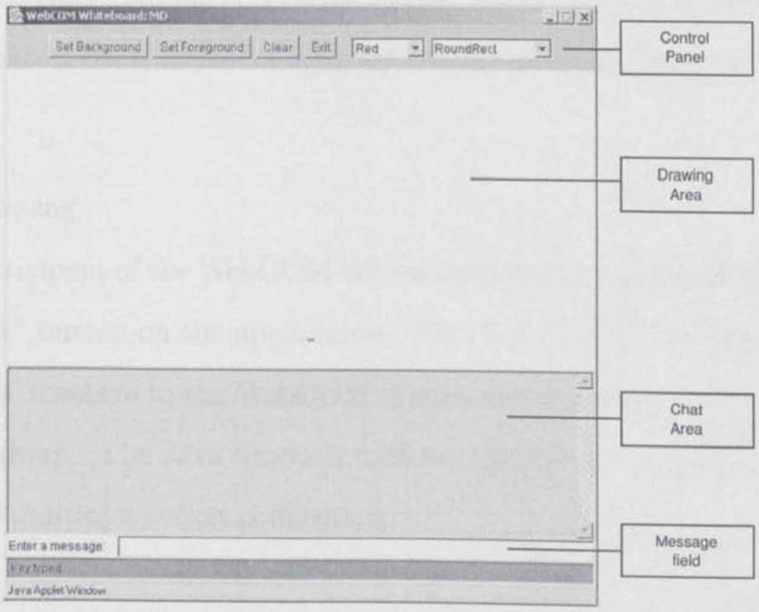


Figure 4.11: WebCOM Whiteboard Interface

Whiteboard System contains four main functions to enable a user to interact with other participants in a group. These functions are:

- Object drawing

  WebCOM Whiteboard provides a basic function to draw on the drawing area as shown in Figure 4.11. The user can select type of object and its attributes from the control panel provided by the application. The object will appear on the drawing area when a user presses and releases their mouse button.

- Data sending

  The WebCOM Whiteboard sends data in UTF8 format. Two types of data are handled by this application, first, data from whiteboard, and second, data from the chatting board. The reason for providing the chatting board in this application is to minimise the use of these two separate services in one application.

- Data receiving

  The WebCOM Whiteboard keeps listening on the multicast socket for any incoming packet. Incoming packets addressed to the recipients are processed internally after being converted into UTF8 data type. Incoming data is processed according to the control code, either as a packet for the drawing board or for the chatting board.

- Group leaving

  Each participant of the WebCOM Whiteboard leaves the joined group by pressing the "Exit" button on the application. This will instruct the application to send the 'leave' message to the WebCOM system and terminate that participant from current group. The Java function used for this process is *socket.close()*, used to close the multicast socket connection.

- Chatting

  The chatting board in this application is simplified from the WebCOM Chat application described previously in Section 4.3.2.

4.4 Web page Design

Figure 4.12 shows the design of Web pages for the WebCOM system prototype. It contains one main page and nine sub-pages. Three of the sub-pages are Java applets that run the WebCOM BBS, WebCOM Chat, and WebCOM Whiteboard. Four of the sub-pages are Java dialog that executed under the BBS system. The main page and the other two pages are constructed using HTML.

4.5 Protocols Flows

Figure 4.13 shows the protocol flows for WebCOM System. Data originating from the application is created. The data generated could be in form of text or number. The first step is to convert the data into UTF8 data format. The converted data is encapsulated

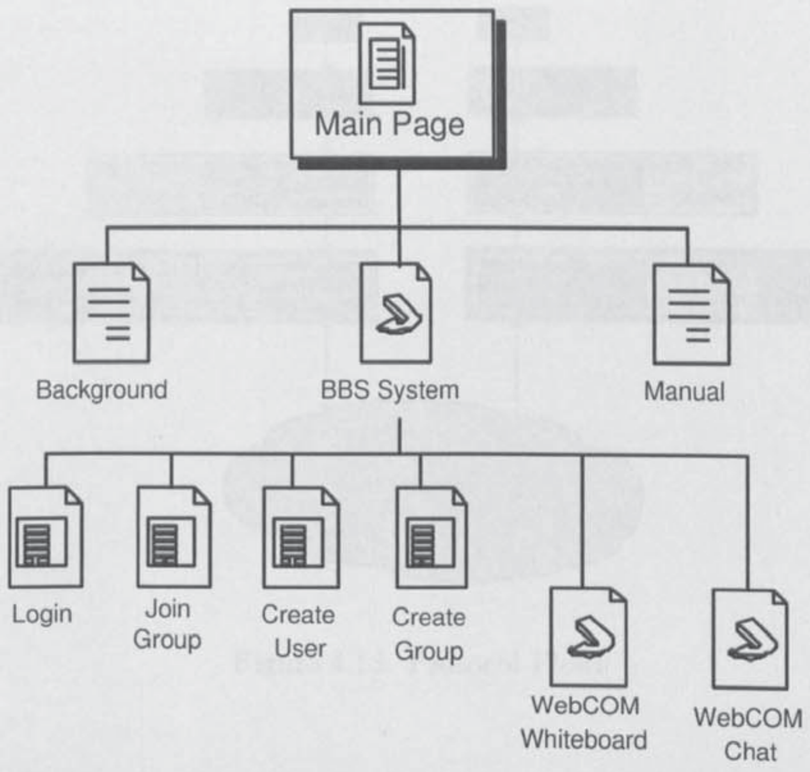


Figure 4.12: Web-pages Design for Prototype

into a packet created using the LRMP profile. The new packet is now ready to be sent out to the network. A multicast socket is used to encapsulate this packet into the multicast socket before it is sent out. After the multicast socket is ready, the socket is encapsulated into Internet Protocol to be send into the Internet.

On the receiver side, the received data is decapsulated until the actual data is extracted by the applications. The opposite processes are implemented on the receiver side which the packet is decapsulated from the raw packet that comes from the Internet into the actual data for the applications.

4.6 Hardware and Software Requirement

Since this application is developed as a platform independent Java applet application, there is no requirement for any specific operating system on which this application can be executed. However all systems require JRE, JDK or JSE to be installed into the

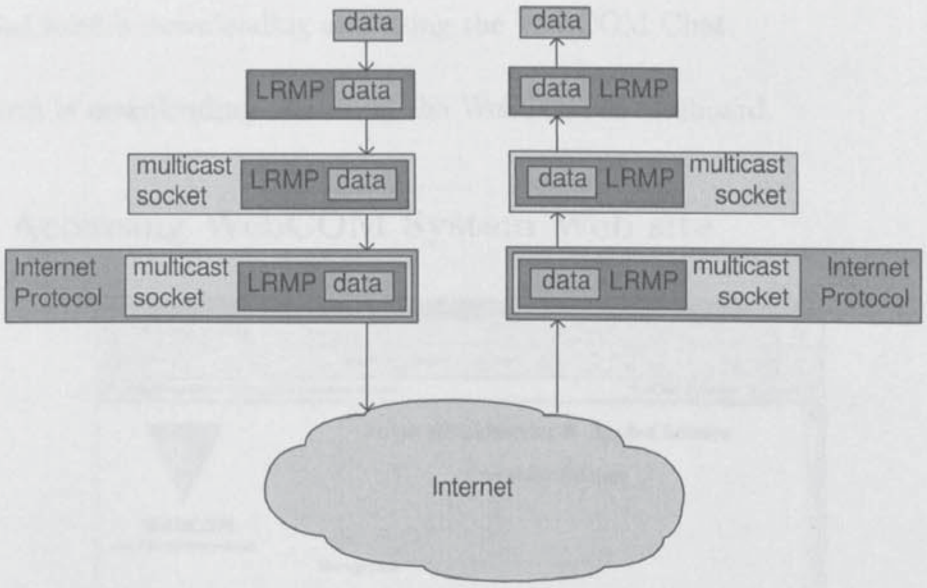


Figure 4.13: Protocol Flows

system, and the user needs a computer that has a network connection to the Internet.

4.7 User Manual

The user manual for using WebCOM System can be found in Appendix I. The manual explains to users how to use the WebCOM System. It starts with how to configure the Web browser to enable the Java applets and then gives further explanation on how to use the CSCW application provided in the WebCOM System.

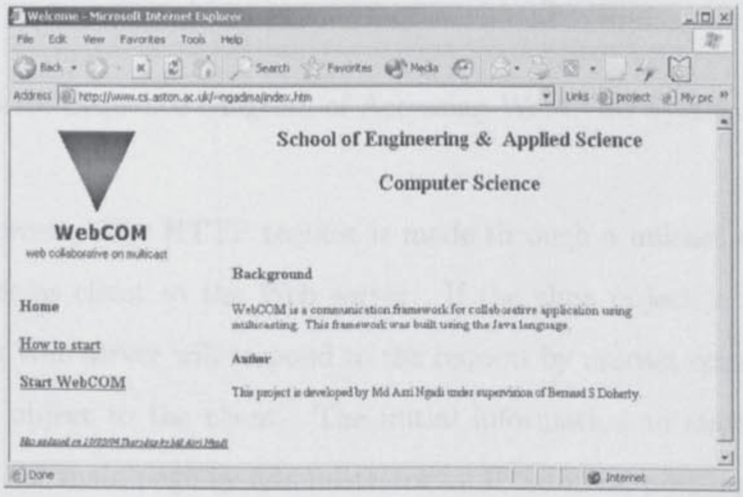
4.8 WebCOM Implementations

This section describes the implementation and operation of WebCOM system. Step by step implementation for each operation is described based on the explanation for each WebCOM component in previous sections. The implementation is divided into four levels of operations:

- First level is accessing the Web sites of WebCOM system.
- Second level is downloading and using the WebCOM BBS.

- Third level is downloading and using the WebCOM Chat.
- Fourth is downloading and using the WebCOM Whiteboard.

4.8.1 Accessing WebCOM System Web site


Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.cs.aston.ac.uk/~ngadima/index.htm
Links project My pic
Done
Internet

School of Engineering & Applied Science

Computer Science

WebCOM
web collaborative on multicast

Background

WebCOM is a communication framework for collaborative application using multicasting. This framework was built using the Java language.

Home

How to start

Author

Start WebCOM

This project is developed by Md Azri Ngadi under supervision of Bernard S Doherty.

[illegible]

Figure 4.14: The Main Page of WebCOM System Website

Users can access the web site of WebCOM system by placing a URL of the system which located on the server side. Figure 4.14 shows the main page of the site. This page contents the information and the manual of WebCOM System. Users can browse the link to get the information. All information is located at Web server and can be accessed from any where as long as the Internet connection is available. This page provides a link to request the WebCOM system. When the link is selected, it will download the Java applet that resides at the Web server and which implements the WebCOM BBS to be executed at client side. The WebCOM BBS is the initial application when the WebCOM system to be started. The sequence diagram for accessing the WebCOM sites is shown in Figure 4.15.

4.8.2 Implementation of WebCOM BBS

The WebCOM BBS is executed on the client side after being completely downloaded from the Web server. The process starts after a user selects the BBS application from

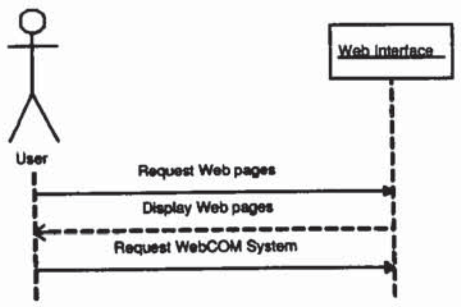
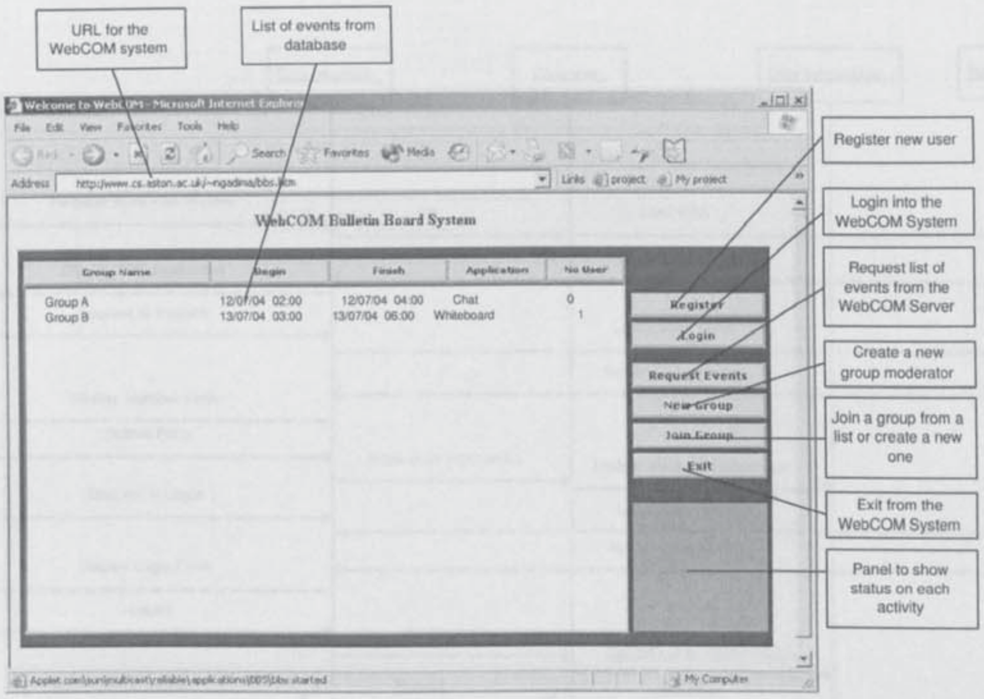


Figure 4.15: Sequence Diagram of Accessing WebCOM System Website

his/her web browser. The HTTP request is made through a unicast connection from the web browser as client to the Web server. If the class object is available in the Web server, the Web server will respond to the request by unicast communication and sends the code object to the client. The initial information to start the WebCOM BBS is given in the main page by administrator i.e IP address, port number, and TTL value. WebCOM BBS uses this information to connect the application to the multicast group in the network. This information will be used also by WebCOM server. In other words, WebCOM BBS and WebCOM Server will be in the same group. If the WebCOM Server is up, operations such as login, register, event request, creating group, and join group can be done. If the WebCOM server is not available all the operations are not functional. Figure 4.16 shows the WebCOM BBS after successful download from the Web server.

After WebCOM BBS is successfully executed on the client side provided WebCOM server is running, all operations shown in Figure 4.16 can be used. All operations between WebCOM BBS and WebCOM server are done by multicast communication. The sequence diagram for all operations in WebCOM BBS is shown in Figure 4.17 and described in Section 3.8.6.

Figure 4.18 shows the communication flow of WebCOM Client BBS and WebCOM Server, this includes internal system communication and multicast communication on the system. The steps are described as follows:


URL for the WebCOM system
List of events from database
Register new user
Login into the WebCOM System
Request list of events from the WebCOM Server
Create a new group moderator
Join a group from a list or create a new one
Exit from the WebCOM System
Panel to show status on each activity
Welcome to WebCOM - Microsoft Internet Explorer
Address http://www.cs.aston.ac.uk/~ngadina/bbs.htm
Register
Login
Request Events
New Group
Join Group
Exit

WebCOM Bulletin Board System

Group Name	Begin	Finish	Application	No User
Group A	12/07/04 02:00	12/07/04 04:00	Chat	0
Group B	13/07/04 03:00	13/07/04 06:00	Whiteboard	1

Figure 4.16: WebCOM BBS

- WebCOM Client BBS requests either to register new user, login to the system, request event from the system, create a new group or join the group. In this stage, WebCOM Client BBS get the information from the end-user through user interface. Then it passed to the WebCOM Communication Module at client side.
- In communication module, this request is converted into UTF8 and put as a byte to the encapsulation process.
- The request byte is encapsulated into a LRMP packet prepared by the system.
- The LRMP packet is sent to a multicast socket that was created during the initial execution of WebCOM Client BBS.
- This packet is multicast to the WebCOM Server.
- WebCOM Communication Module at server side receives this packet.
- The server communication module then receives the packet from the multicast socket.

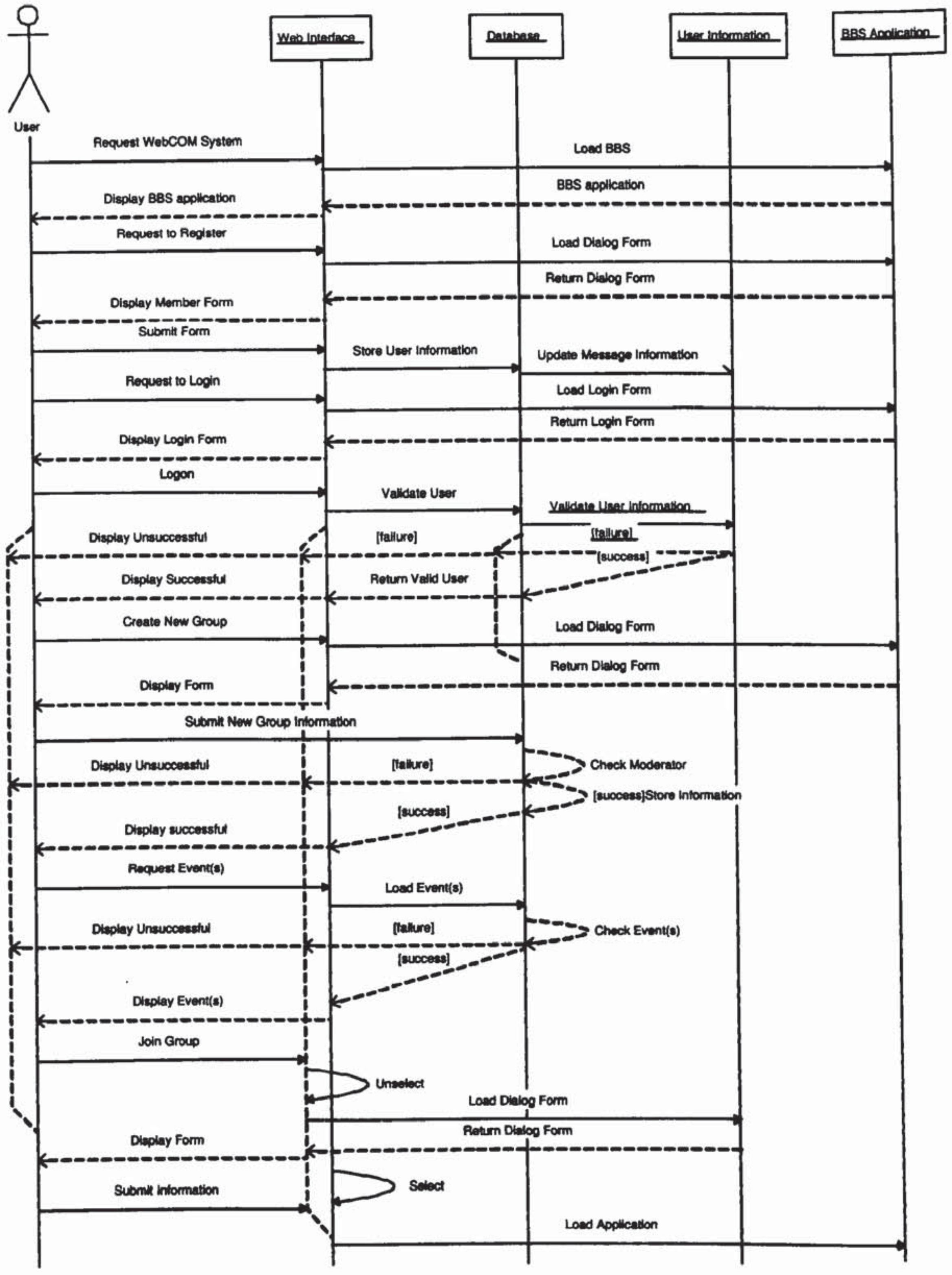


Figure 4.17: Sequence Diagram of All Operation in WebCOM BBS

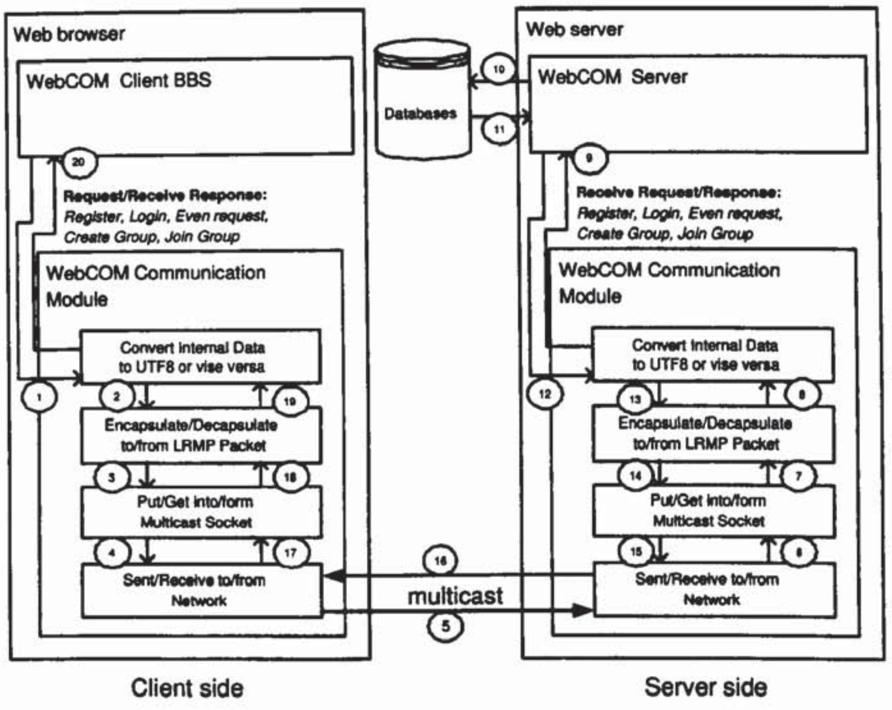

Web browser
WebCOM Client BBS
Request/Receive Response:
Register, Login, Even request, Create Group, Join Group
WebCOM Communication Module
Convert internal Data to UTF8 or vise versa
Encapsulate/Decapsulate to/from LRMP Packet
Put/Get into/form Multicast Socket
Sent/Receive to/from Network
Databases
Web server
WebCOM Server
Receive Request/Response:
Register, Login, Even request, Create Group, Join Group
multicast
Client side
Server side

Figure 4.18: Communication Flow of the WebCOM BBS

- The packet is decapsulated into byte form.
- The byte form is converted from UTF8 into internal data type.
- This data is passed to the WebCOM Server to process it.
- If the process is required to access the databases, it will send the request to the databases,
- and the databases responds to the WebCOM Server request.
- The response is passed to the WebCOM Communication module.
- The response is convert to the UTF8 and put as a byte to the encapsulation process.
- This byte in encapsulated into LRMP packet that prepared by the system.
- This packet is put into the multicast socket that created during the initial execution of WebCOM Client BBS.

- This packet is multicast to the WebCOM Client BBS.
- WebCOM Communication Module at client side receives this packet.
- The client communication module then get the packet from the multicast socket.
- The packet is decapsulated into byte form.
- The byte form is converted from UTF8 into internal data type.
- This data is passed to the WebCOM Client BBS to process it.

4.8.3 Implementation of WebCOM Chat/WebCOM Whiteboard

WebCOM Client Chat and WebCOM Client Whiteboard are created from WebCOM Client BBS as shown in Figure 4.19. WebCOM BBS provides all information about groups that available in the WebCOM system. When end-user select to join certain group in WebCOM system, WebCOM BBS will create an application that belongs to that group. After WebCOM client is created, there will be no more connection between WebCOM BBS and WebCOM clients. The connection will be totally on WebCOM clients only and the communication is done with multicasting.

Figure 4.18 shows the communication flow of WebCOM clients either between WebCOM Client Chat or WebCOM Client Whiteboard. The communication between WebCOM clients is done using multicasting. The step is described as follow:

- WebCOM client sends data either texts of drawings. WebCOM clients get the information from the end-user through user interface. The data is passed to the WebCOM Communication Module.
- In communication module, this data is converted into UTF8 and put as a byte to the encapsulation process.
- This byte in encapsulated into LRMP packet prepared by the system.

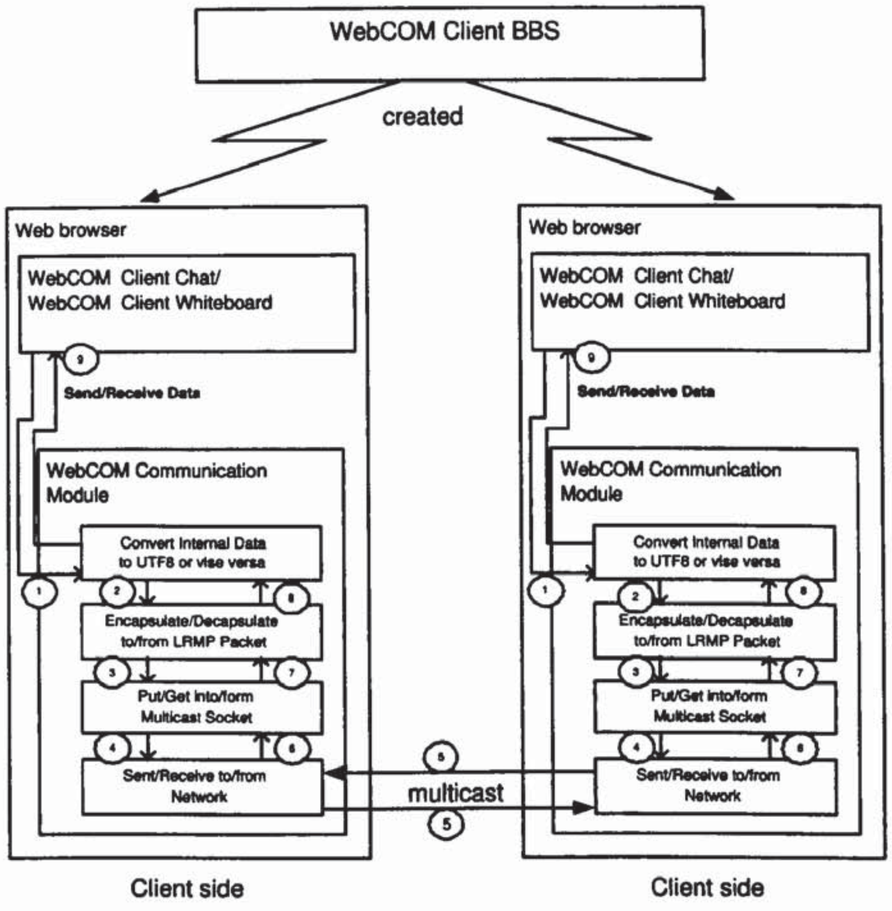

WebCOM Client BBS
created
Web browser
WebCOM Client Chat/
WebCOM Client Whiteboard
Send/Receive Data
WebCOM Communication Module
Convert Internal Data to UTF8 or vise versa
Encapsulate/Decapsulate to/from LRMP Packet
Put/Get into/form Multicast Socket
Sent/Receive to/from Network
multicast
Client side
Client side

Figure 4.19: Flow of the WebCOM Client Chat and Whiteboard

- This packet is put into the multicast socket created during the initial execution of WebCOM clients.
- This packet is multicast to other WebCOM clients.
- WebCOM Communication Module at other client side receives this packet.
- The communication module then gets the packet from the multicast socket.
- The packet is decapsulated into byte form.
- The byte form is converted from UTF8 into internal data type.
- This data is passed to the WebCOM client application to be processed by it.

- These processes are followed by other clients who want responses to the received data.

Chapter 5

Experiment, Results and Analysis

"The best tools for large-scale Internet collaboration are those that effectively minimize actual use of the Internet... " (Fielding and Kaiser, 1997)

5.1 Introduction

This chapter describes the testing process and outcome of the testing. The work in this thesis has divided the experiment into performance and functionality experiments. Performance experiments refer to the testing of network performance on a testbed, while the functionality experiment refers to the testing of the ability of the communication framework, WebCOM, to support certain selected features. The purpose of the testing is to evaluate the performance of WebCOM implementing CSCW applications, and to evaluate the ability of the WebCOM to support CSCW applications. The experimental testing consists of two parts. In the first part, the experiment involved a prototype that runs on WebCOM. In the second part, the experiment involved a prototype that runs on a client/server unicasting model.

5.2 Experiment of Functionality

This section describes functionality testing on WebCOM. The functionality experiment in this thesis refers to the tests on accessibility and portability of WebCOM which does not require performance measurements. These features are tested only on WebCOM and not on the system that uses unicasting, because these aspects are already well known in the client/server model.

5.3 Experiment of Performance

The testing was done in a LAN environment connected to the Internet. Twenty seven PCs were used to represent the participants in a group session, because :

- only 27 machines are available
- although there is no specific ideal number of users for a group, twenty seven is large enough to represent a significant group of users. Gmez *et al.* (2001) allocated two users in their applications and Benbunan-Fich *et al.* (2003) allocated six or seven in a group for their applications.

Every user is connected to each other and is able to access any site or computer in the Internet. From this environment a testbed is set up for experiment.

5.3.1 Testbed

Figure 5.1 shows a network architecture of a Computer Science laboratory used as a testbed for the prototype. The architecture uses the existing network configuration of the Computer Science laboratory without any change. The lab consists of 27 PCs and each PC is equipped with a 750MHz Pentium 3 processor and 128 megabytes of RAM. The PCs in the lab are interconnected using three Ethernet switches. Each PC has a dedicated 10 Mbps connection to the switch. From the switches, the lab is interconnected to outside computers including a server and the Internet. The prototype is stored in a server which is connected to the Internet. All PCs run the Windows XP operating system and the server runs the Unix operating system. A Java-enabled Web browser is installed in all PCs to enable users to use the prototype system.

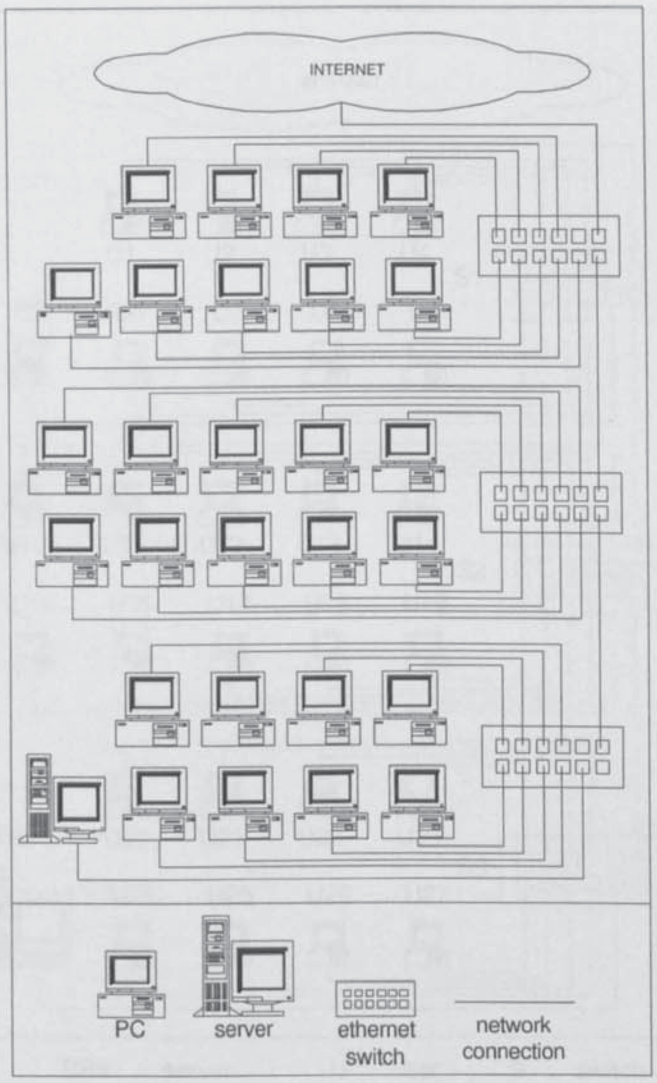


Figure 5.1: The Architecture of a LAN Used as Testbed

5.3.2 Testing approach

Each PC in the lab is assigned to a single user. In order to perform experiments, each PC is numbered from 1 to 27, as shown in Figure 5.2. The experiment was done with groups of 3, 6, 9 up to 27 users. There are two ways that users for a group were selected:

- users chosen according their number. If the experiment was for a group of three users, PCs numbered 1, 2, and 3 were used, if the experiment was for a group of twelve users, PCs numbered 1 up to 12 were used, and so on.

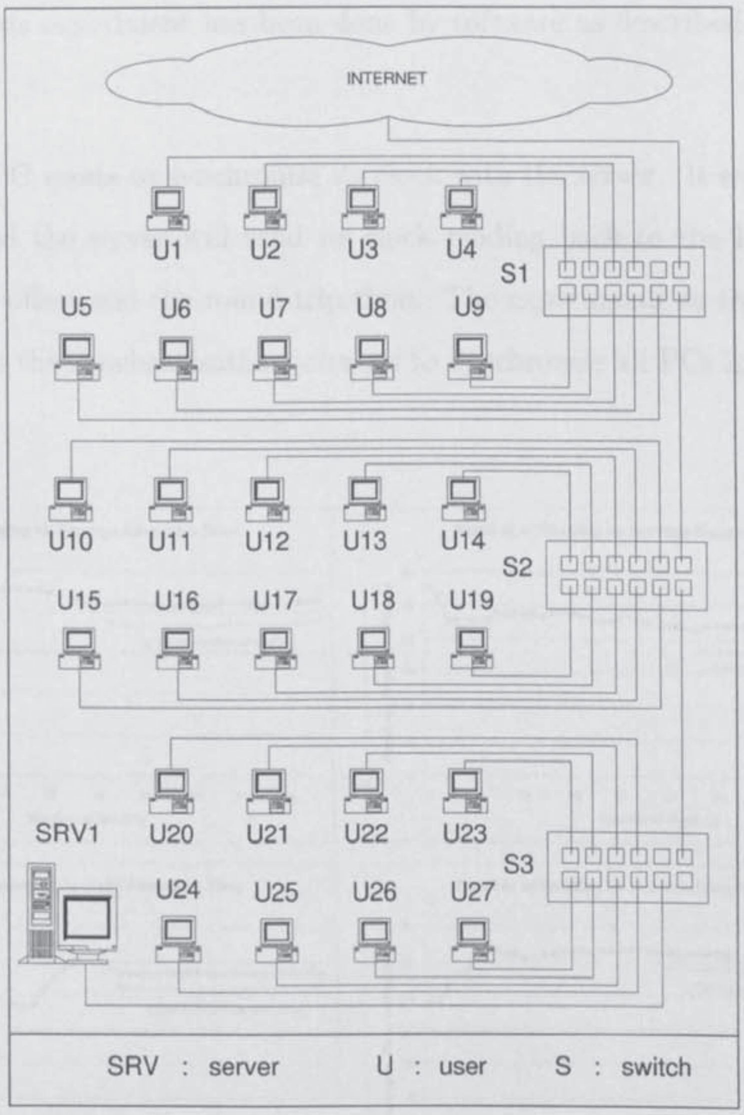


Figure 5.2: The Arrangement of PC in the Testbed

- users chosen according to their position in their switch. If the experiment was for a group of three users, PCs numbered 1, 10, and 20 were used, which is the first PC from each switch, to remove the effect of localisation under a switch from the experiment.

Before any experiment is started, an important and critical part of the testing in this study is time synchronisation. To calculate the performance metrics accurately, PC clocks must be synchronised. Clock synchronisation can be achieved by software synchronisation, hardware synchronisation, or a combination of the two. Clock synchronisation in this experiment has been done by software as described in Appendix G.2.

Suppose the PC wants to synchronise its clock with the server. It sends a request to the server, and the server will send its clock reading back to the PC. The PC records the clock offset and the round trip time. The experiments in this thesis used *SP TimeSync*[1] as the synchronisation software to synchronise all PCs involved in the experiment.

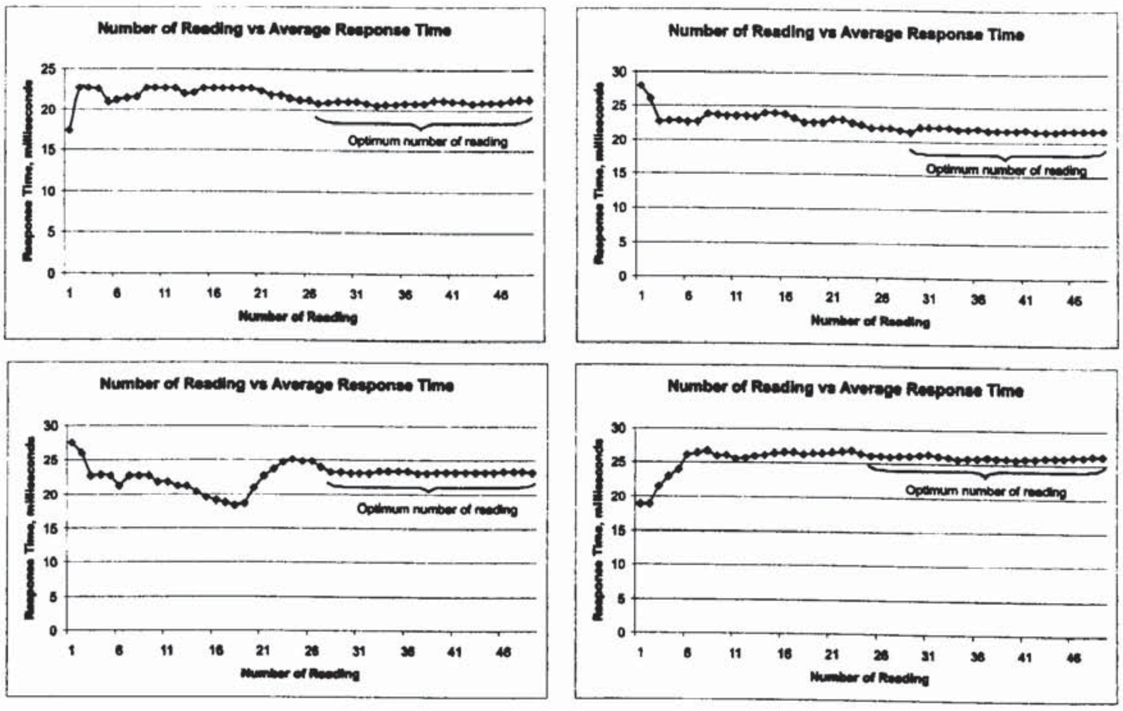


Figure 5.3: The Optimal Number of Readings

The experiments are divided into two parts. In the first part, the prototype runs on top of WebCOM and the second part, the prototype runs on a client/server model using unicasting. All experiments are run on the same testbed, using the same method of testing and using the same performance metrics. The experiment is done by sending a sequence of data of different sizes from a user in a group as sender to other users in a same group as receiver. When the experiment is done for a group of users, every user

[1]refer Appendix G.2

in that group is required to send a sequence of testing data to all other users in a same group so all are equal participants in the group. The final results are taken from the average of readings for every user in that group.

All the performance measurements were measured in milliseconds using the system clock[2]. Each measurement was repeated forty times and the results averaged for accuracy. Experiment showed that the average changed very little after 40 readings as shown in Figure 5.3.

5.4 Performance Metrics

Performance metrics are used to determine the performance measurement. In this thesis three metrics of network performance are considered to measure the efficiency of WebCOM, as explained in Chapter 2, Section 2.8.2. These metrics are chosen because these are the metrics most used in network performance measurement (Karger *et al.*, 1999; Arlitt *et al.*, 2000; Liu *et al.*, 2001). Most of the authors took two important metrics in their study such as throughput and overhead (Hong *et al.*, 1999), packet loss and round-trip loss (Borella *et al.*, 1998), throughput and latency (Ahmed *et al.*, 2002), and delay and packet loss (Barford and Crovella, 1999). The work in this thesis has chosen three performance metrics to evaluate the performance of WebCOM. These metrics are:

- **Response time**

  Response time is the amount of time it takes a packet to travel from source to destination specifically from the submission end-point on the source to the beginning of a response at receiver. The response time includes the transmission time and the processing time. Sometimes the response time is called latency.

- **Packet loss**

  Packet loss is defined as the number of packets lost while traveling on the wire

[2]Appendix 2.8.3 and G

through the network from sender to receiver. Packet loss is measured either by the number lost or by calculating percentage of packets lost compared to the number of packets sent by the sender.

- **Throughput**

  This is the amount of data transferred in one direction over a link divided by the time taken to transfer it. It is usually expressed in bits per second (bps) or bytes per second (Bps).

5.5 Performance Parameters

Many factors that might effect network performance of a system have been reported, such as data size (Klovning and Bonaventure, 1996), group size and session length (Tseng *et al.*, 1999), and file size and load of network (Barford and Crovella, 1999). The work in this thesis is not intended to cover all the factors that might effect WebCOM. However, the factors considered important in this work have been described in the previous section (Section 5.4). Furthermore, these factors are variable and not fixed, thus giving flexibility to make adjustment to the factors depending on system requirement.

In order to measure the network performance metrics, two factors/parameters are considered for the experiment:

- **Size of Packet**

  There are many forms of data moving on the Internet such as text, graphics, image, voice, and video stream. All these forms of data are transmitted in packets for transmission over the network. Generally, the size of packet transmitted on the network will affect the network performance (Barford and Crovella, 1999). This parameter was considered because it is a main parameter to test the response time, packet loss and throughput of WebCOM.

  The largest non-fragmenting packet (MTU = Maximum Transfer Unit) depends on the networks involved. For Ethernet, the size of packet is up to 1500 bytes including a margin for IP headers. Most high speed networks have larger maximum data size for example ATM[3]. For experiments done in this thesis, which used Ethernet to run the experiments, the maximum data size can be handle is up to 1500 bytes.

The LRMP places an upper bound on the maximum length of packets that can be transmitted, i.e. the maximum transmission unit(MTU), which is set to 1400 bytes. So, the supplier generated 1384 bytes messages, plus 1 byte header and 15 bytes LRMP header, in a total of 1400 bytes at a fixed rate (target throughput). An empty packet sent by LRMP over the network is 40 bytes[4]. The size of packet used for the experiment varies, depending on the performance metrics. For response time and throughput measurement the packet size (including the packet header) is from 40 bytes up to 1424 bytes, and for packet loss is 1424 bytes.

- **Number of receivers**

  The objectives of using groupware application are achieved by having more than one participant in each discussion. The number of participants vary depending on the group. The work in this thesis, considers up to 27 users for each experiment. This number is explained in Section 5.3.2.

All parameters and performance metrics used to measure the performance of WebCOM were also used for a system running on client/server unicasting.

5.6 Result on Experimental testing

This section presents experimental results for the WebCOM System running on a testbed. The tests measured three performance metrics of WebCOM: response time,

[3]Refer to Chapter 2 Section 2.5.2

[4]Refer to Appendix F.4.1

throughput, and packet loss. These metrics have been considered in order to evaluate the effectiveness of WebCOM. Another experiment used in this thesis is an application that runs on client/server unicasting model. This system was used for comparison with the WebCOM System.

The performance metrics were:

- average response time for WebCOM and "normal system"
- effect of user numbers on response time for WebCOM and "normal system"
- effect of size of packet on response time for WebCOM and "normal system"
- effect of user numbers on throughput for WebCOM
- effect of user numbers on packet loss for WebCOM

5.6.1 Response Time for WebCOM

As mentioned in Section 5.4, response time was defined as an elapsed time between the end of an inquiry or demand on a computer system/sender and the beginning of a response at the receiver. The response time includes the transmission time, and the processing time. The synonyms for response time are latency, latent period, and reaction time.

Figure 5.4 shows the average response time for WebCOM. The figure shows that the response times appear to generally increase as the number of users in a group increased from 3 to 27 users. Figure 5.4 shows one point which lies away from the general trend; this is thought to be caused by the switches in the network configuration.

Effect of User Number

Figure 5.5, Figure 5.6, and Figure 5.7 show the effect of user numbers on response time for three different users, namely, User 1, User 10, and User 20 respectively. From these graphs, the response time appears to grow approximately linearly when the number of

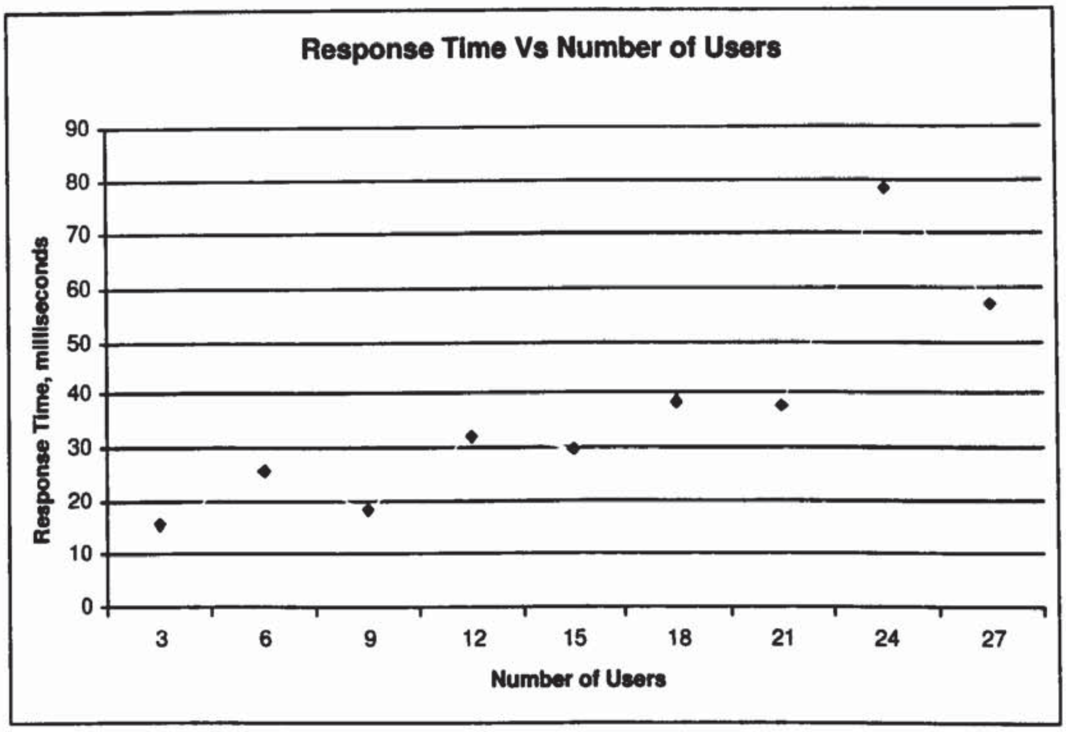


Figure 5.4: The Average Response Time for WebCOM

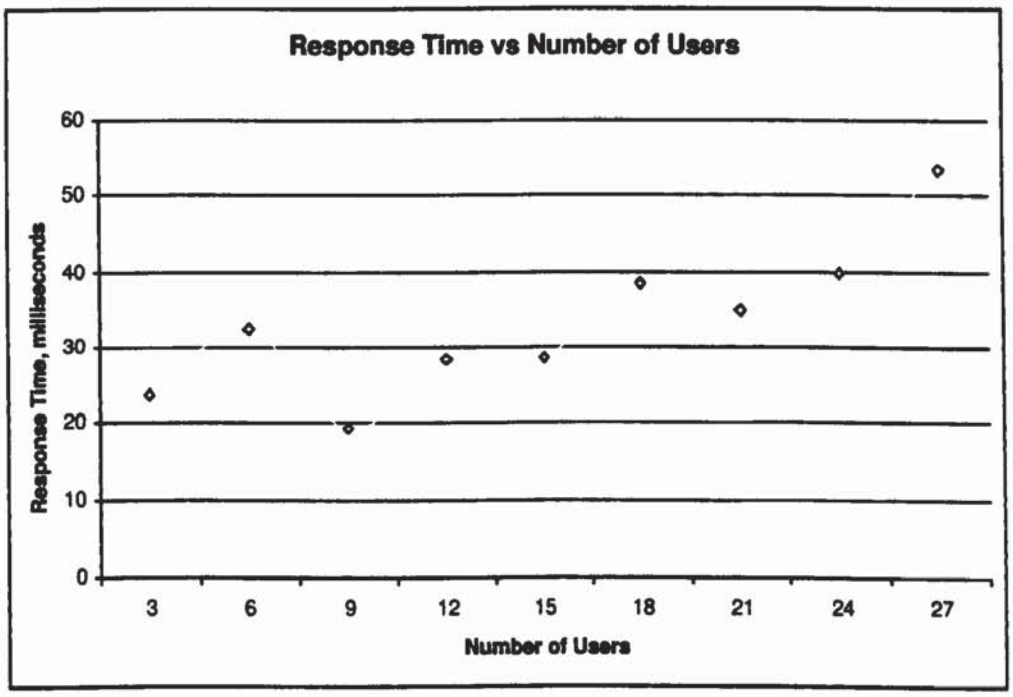


Figure 5.5: The Effect Number of Users on Average Response Time for User 1

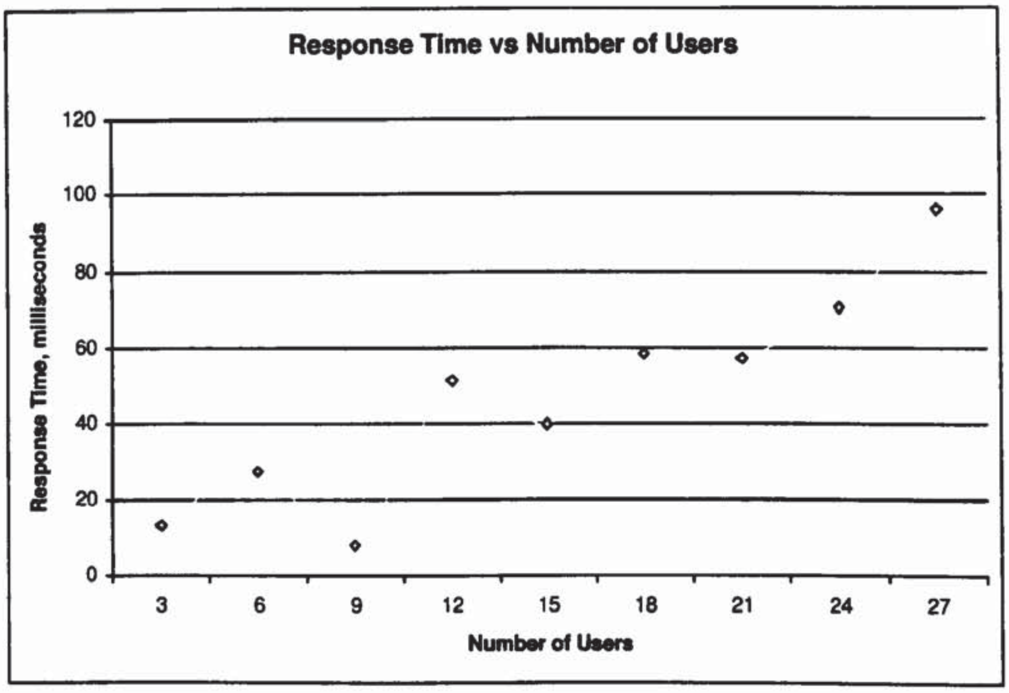


Figure 5.6: The Effect Number of Users on Average Response Time for User 10

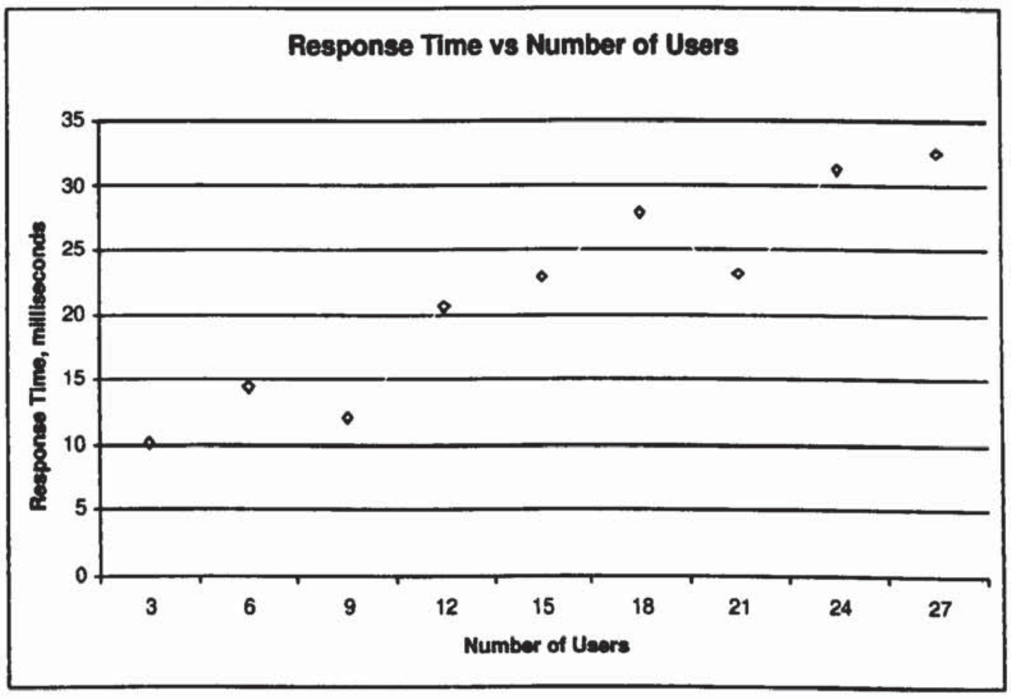


Figure 5.7: The Effect Number of Users on Average Response Time for User 20

users in a group increased from 3 to 27 users. This is in accord with the theoretical concept where normally the response time will increase if the number of user that participate in a collaboration increases. The affect of running the WebCOM system in a LAN also makes the behavior of the system similar to the unicasting, however with slightly better performance. This result will appear in following sections.

Effect Size of Packet

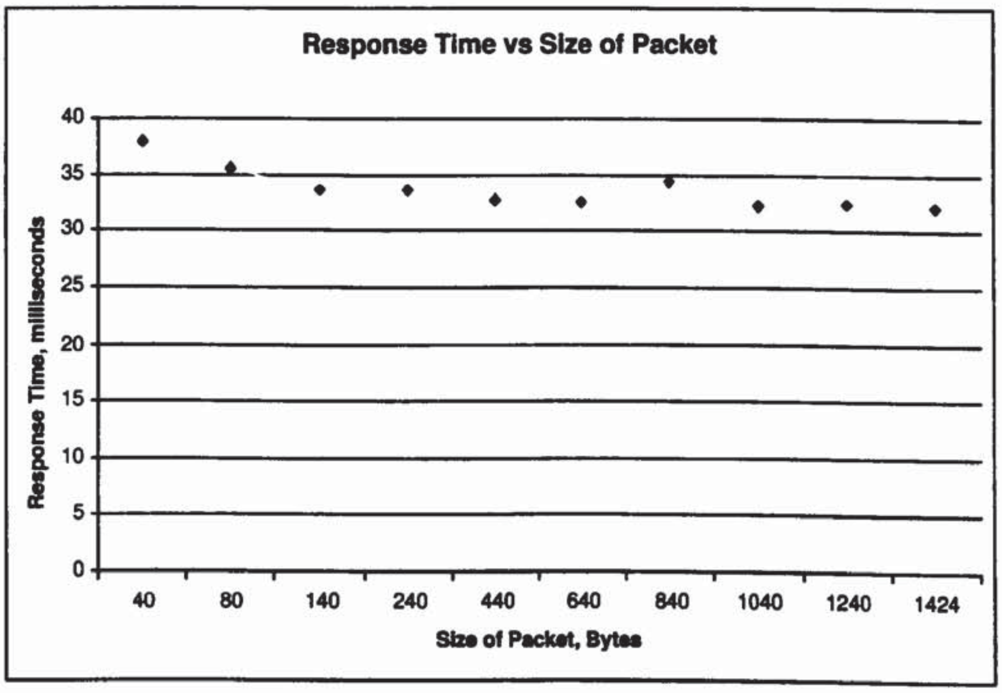


Figure 5.8: The Effect Size of Packet on Average Response Time for User 1

Figure 5.8, Figure 5.9, and 5.10 show the effect of packet size sent from a sender to all users in the group, with response time measured for three different users, namely, User 1, User 10, and User 20 respectively. From these graphs, the response time declines slightly the size of a packet used to carry data increased from 40 to 1424 bytes. This is in accord with the theoretical concept where normally the response time will decline when the size of packet is increases, due to the proportionately lower overhead with a larger packet.

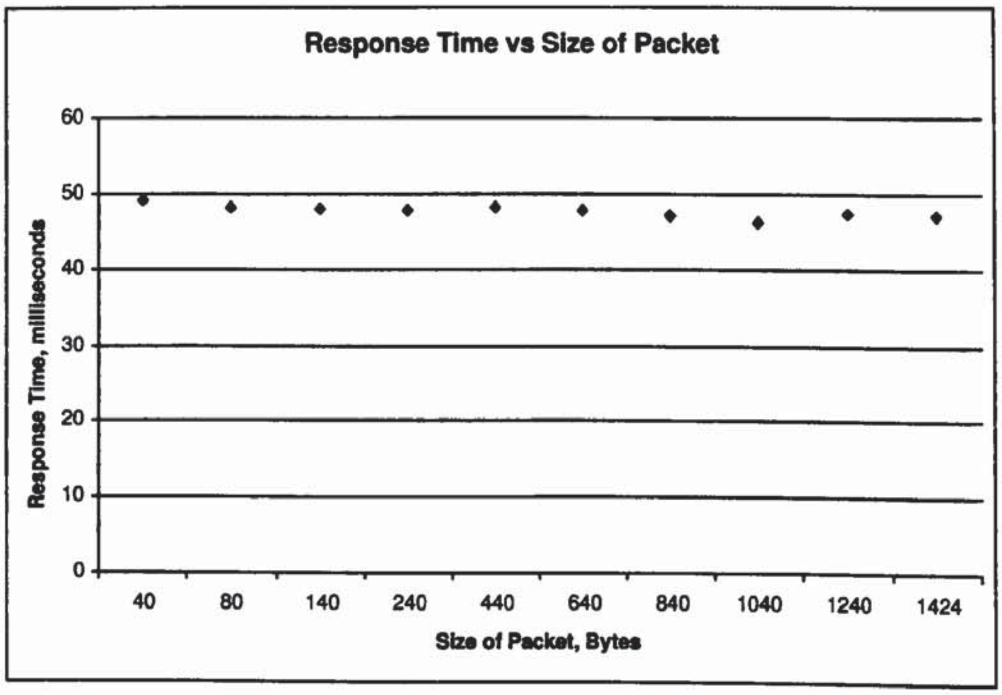


Figure 5.9: The Effect Size of Packet on Average Response Time for User 10

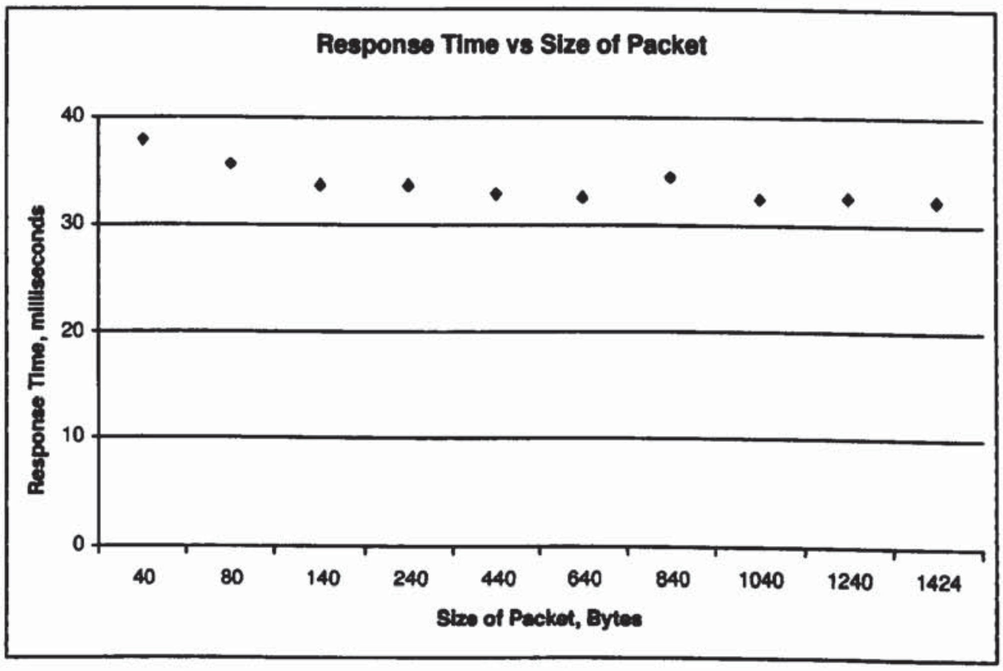


Figure 5.10: The Effect Size of Packet on Average Response Time for User 20

5.6.2 Throughput for WebCOM

As mentioned in Section 5.4, throughput is defined as the amount of data transferred in one direction over a link divided by the time taken to transfer it. It is usually expressed in bits or bytes per second. Throughput is calculated as the data size divided by the transfer time (Wolski, 1998).

$$\textit{throughput} = \textit{data size} /(\textit{data transfer time})$$

Figure 5.11, Figure 5.12, and Figure 5.13 show the throughput for three different users in a group, namely, User 1, User 10, and User 20 respectively. The figures show the throughput for User 1, User 10, and User 20 appears broadly to decay when the number of users increases from 3 to 27 users. This event suggests an advantage for WebCOM where throughput for the group will decrease as the number of users increases. Figure 5.11 and 5.12 show one point which lies away from the trend of points when the number of users is up to nine users; this is thought to be caused by the switches in the network configuration and buffering system at end-user/client machines.

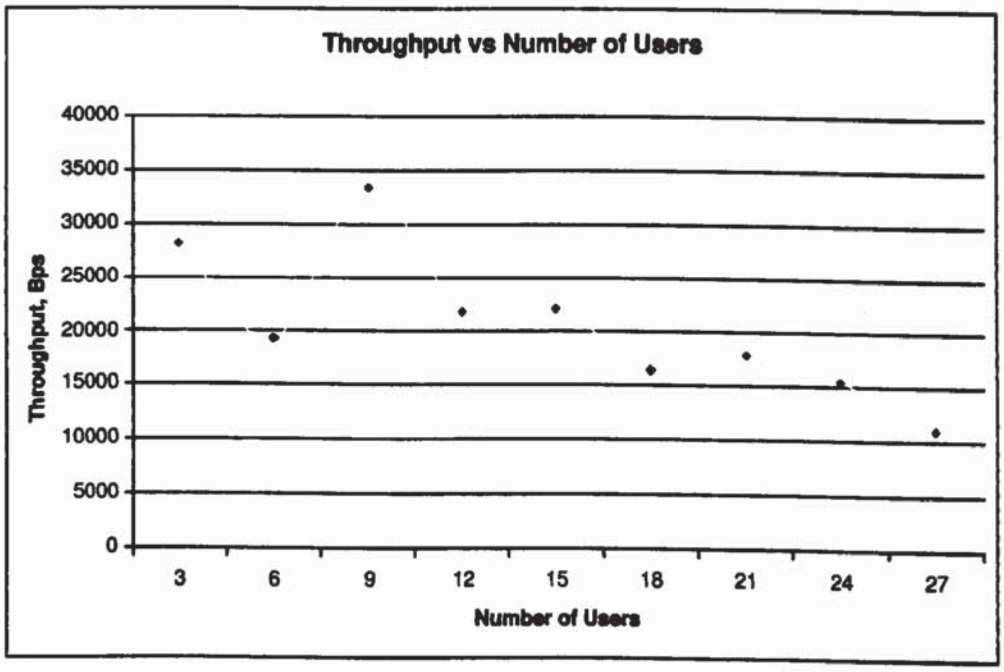


Figure 5.11: Throughput at User 1

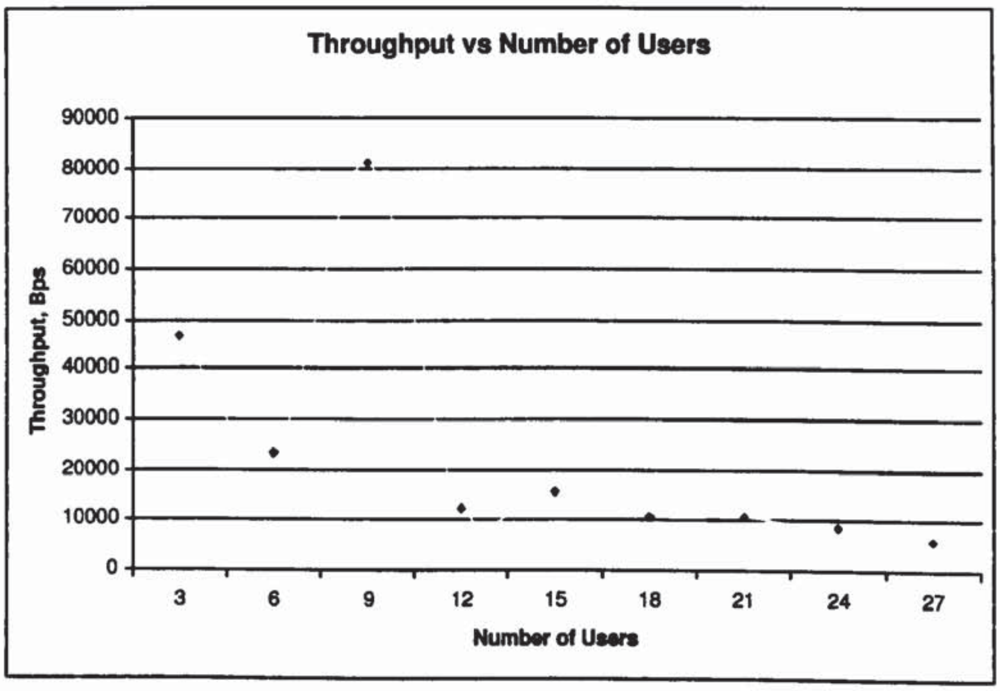


Figure 5.12: Throughput at User 10

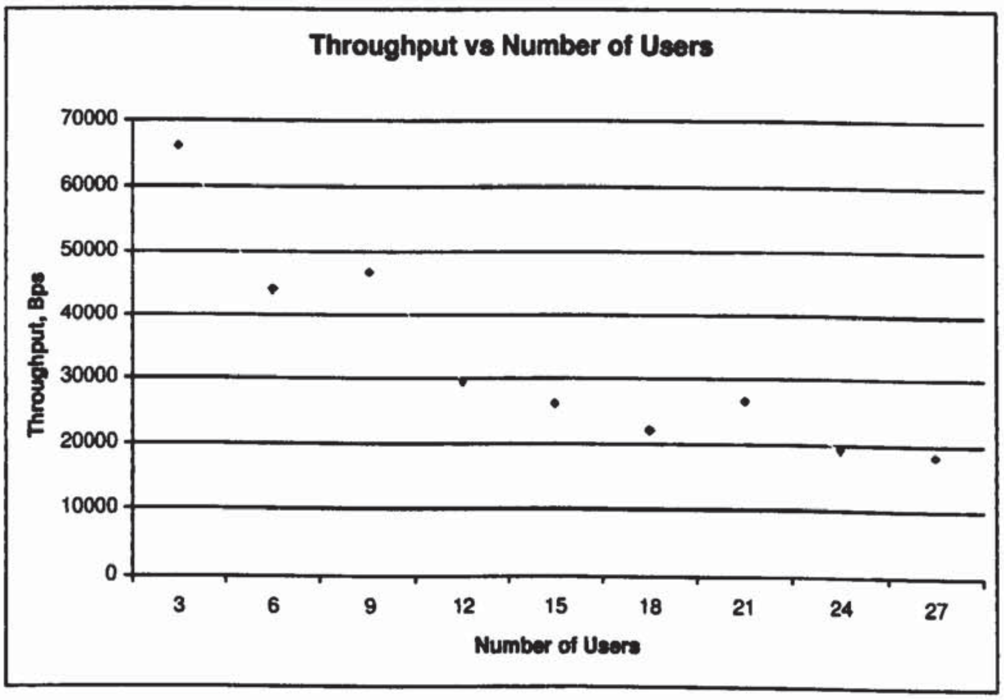


Figure 5.13: Throughput at User 20

5.6.3 Packet Loss for WebCOM

Packet loss is measured either by the number of packets lost or by calculating the percentage of packets lost compared to the number of packets sent by the sender. This test was run for approximately two hours with packet size 1384 bytes in each packet. Twenty seven users were used for the testing. The number of packets sent within two hours was 32651 packets. The sender sent the packets continuously within that time.

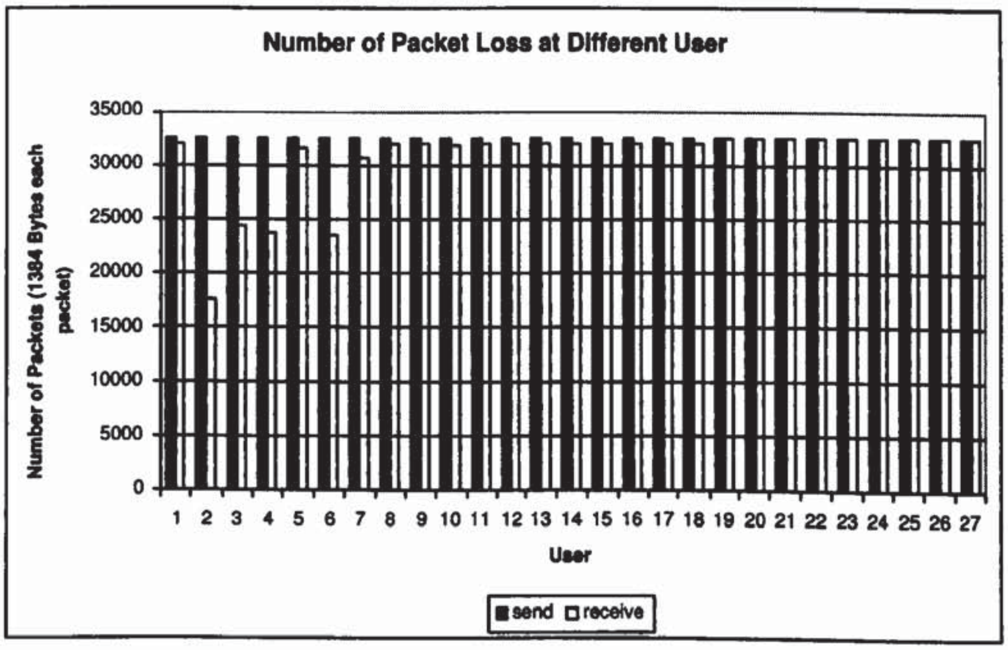


Figure 5.14: Number of Packet Loss For Each User Within Two Hours

Figure 5.14 shows the number of packets lost during transmission for each user within a group. It shows that some users received packets without loss, some of them had a few lost, and some of them had many lost. As shown in Figure 5.15, User 1 and User 8 up to User 18, the percentage of packet loss is 1.7%. User 19 up to User 27 receive all packets without error. User 2, 3, 4, 5, 6, and 7 experience a varying percentage of lost packets: 46.2%, 25.4%, 27.3%, 2.9%, 28.1%, and 5.4% respectively.

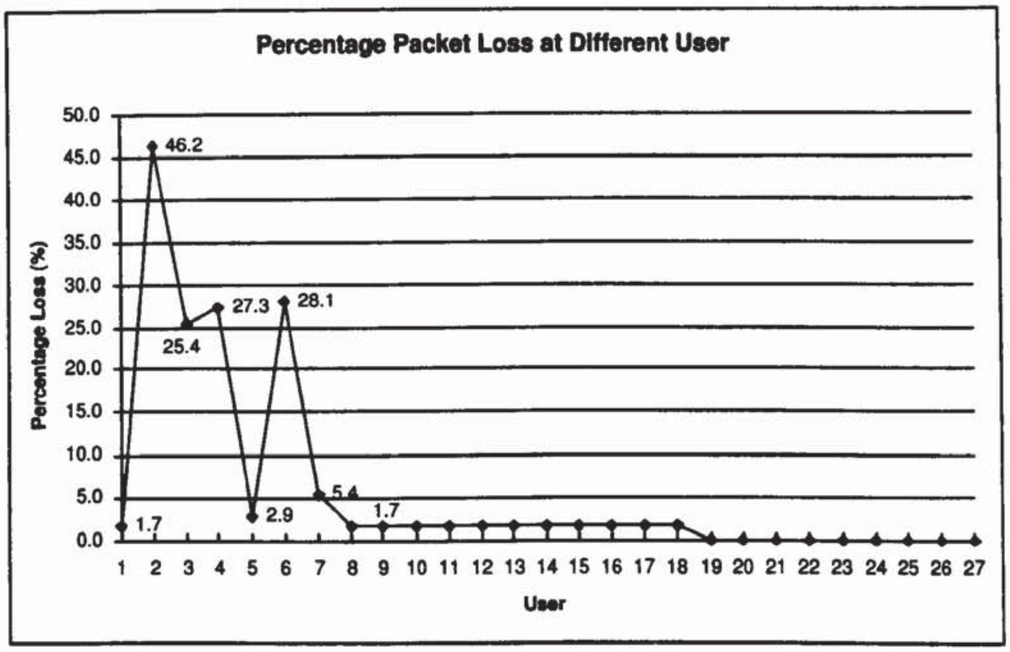


Figure 5.15: Percentage of Packet Loss Within Two Hours

5.7 Result for Normal System

This section presents results for a "normal system" that runs on a client/server unicasting model. The system was built for comparison with WebCOM.

5.7.1 Response Time for Normal System

This subsection presents the effect of user numbers and size of packet on average response time for a "normal system".

Effect of Number of User

Figure 5.16 shows the average response time for the "normal system" using client/server unicasting. The response time appears to increase approximately linearly from 190 to 300 milliseconds when the number of users increases from 3 to 27 users.

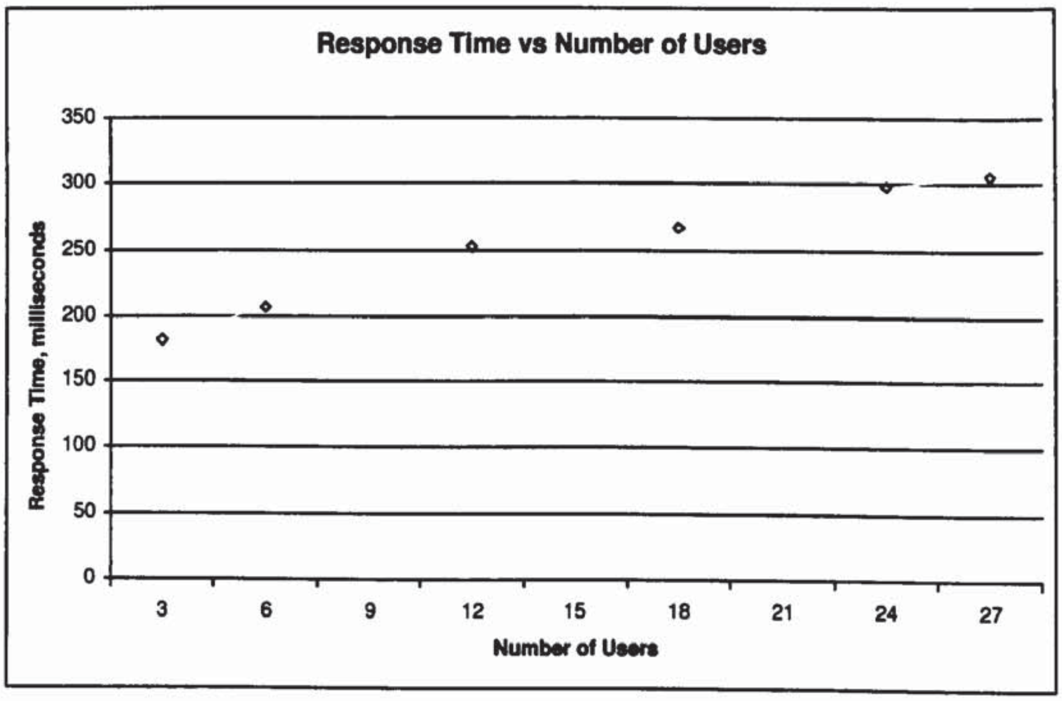


Figure 5.16: The Effect Number of Users On Response Time

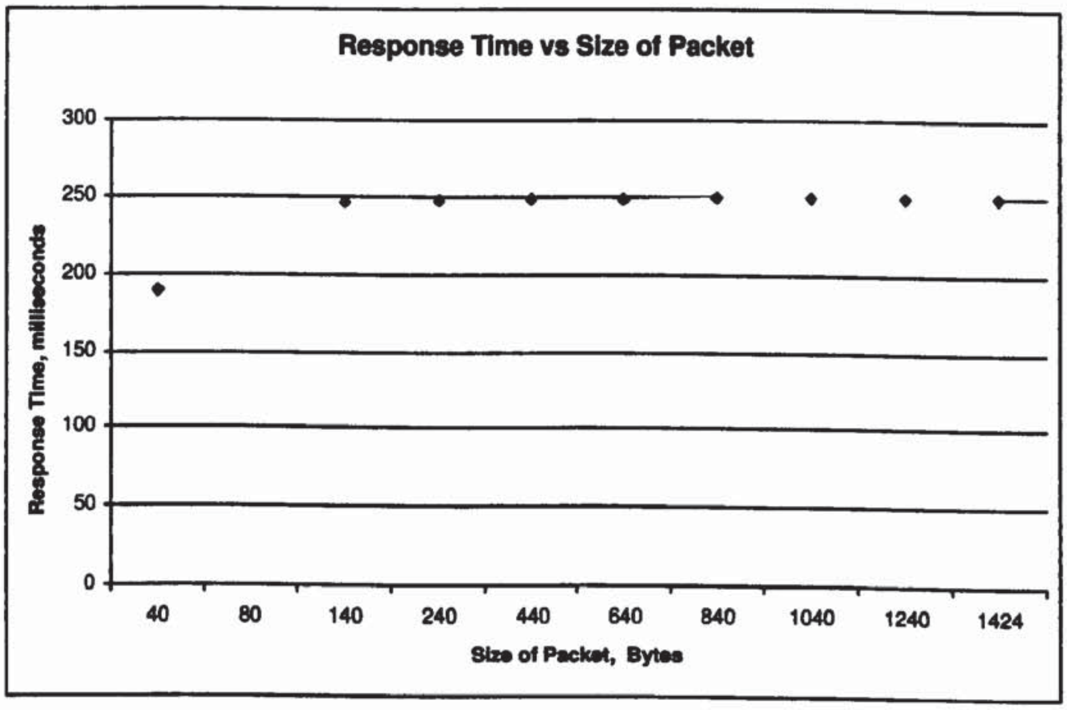


Figure 5.17: The Effect Size of Packet on Response Time

Effect of Size of Packet

Figure 5.17 shows the average of response times for the "normal system" when the data size per packet increases from 40 to 1424 bytes. It shows that the response time appears to increase approximately polynomially from the beginning, then stop increasing when the response time reached 250 milliseconds. It stopped increasing when the size of packet was between 140 to 1424 bytes.

5.7.2 Throughput for Normal System

This subsection presents the effect of user numbers and size of packet on average throughput for a "normal system".

Effect of User Numbers

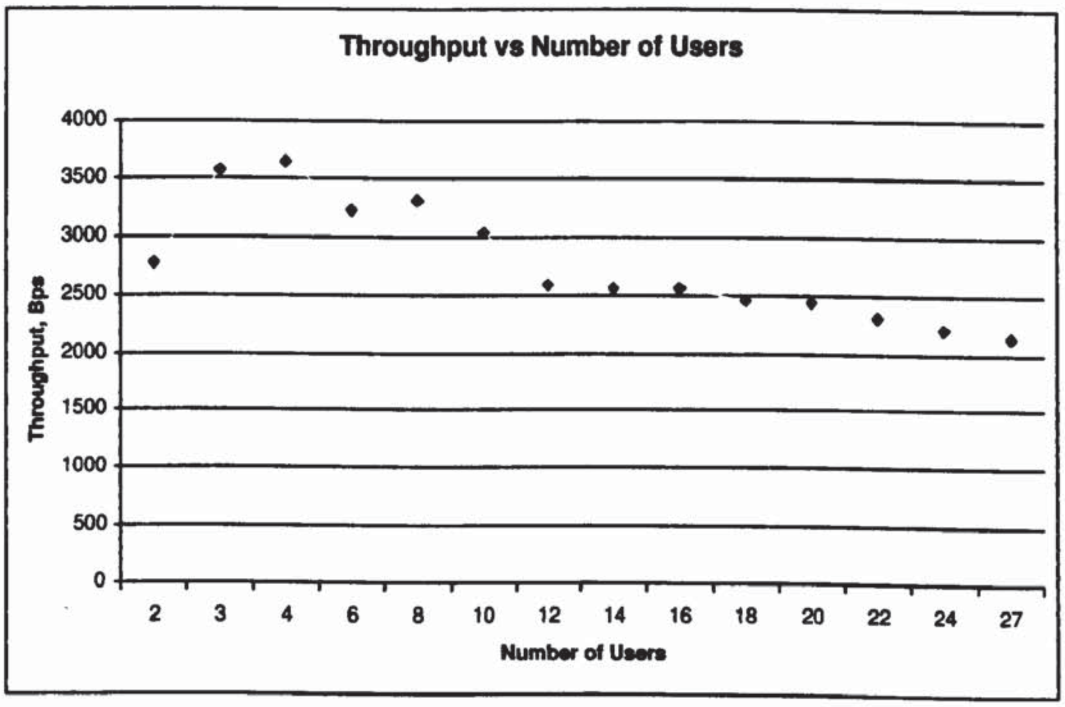


Figure 5.18: The Effect Number of Users On Throughput

Figure 5.18 shows the average throughput for the "normal system" using client/server unicasting. The throughput decays approximately linearly when the number of users

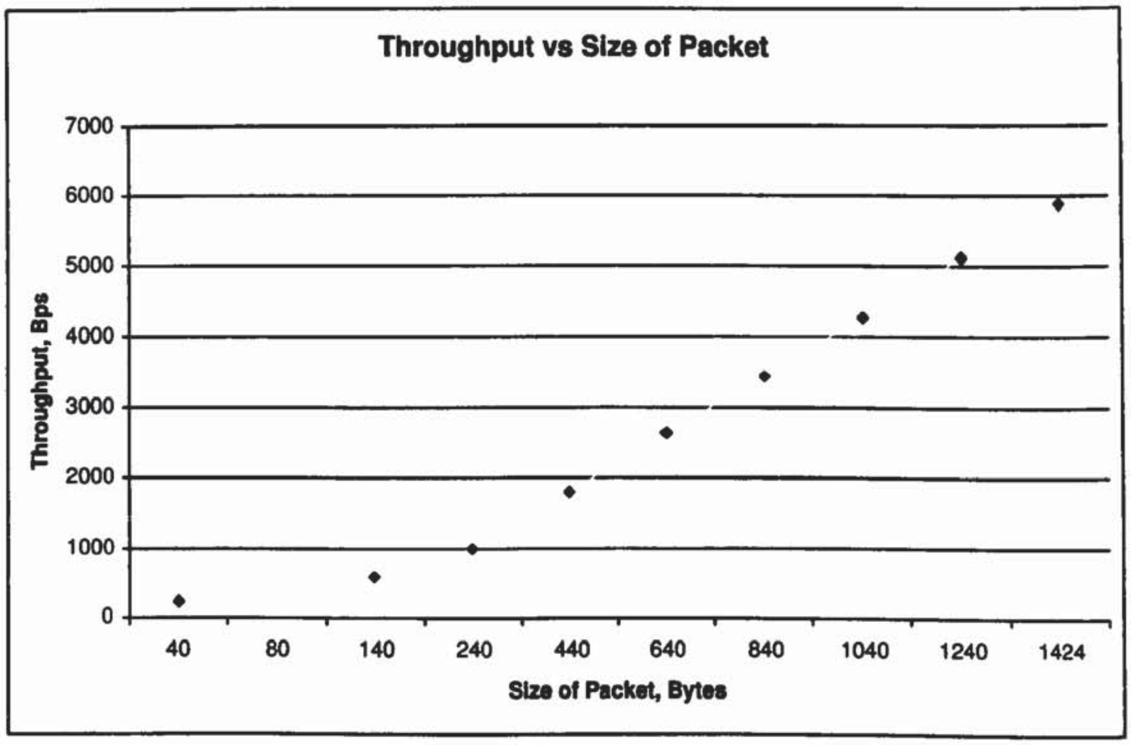


Figure 5.19: The Effect Size of Packet on Throughput

increases from 3 to 27 users.

Effect Size of Packet

Figure 5.19 shows the average throughput for the "normal system" when the data size per packet increases from 40 to 1424 bytes. It shows that the throughput appears to increase approximately as a power law when the packet size is between 40 and 240 bytes. When the packet size above 240 bytes, the throughput appears to increase approximately linearly.

5.8 Comparison WebCOM and Normal System

This section compares the metrics of network performance between WebCOM and the "normal system". Analysis and further discussion on each result is also presented here.

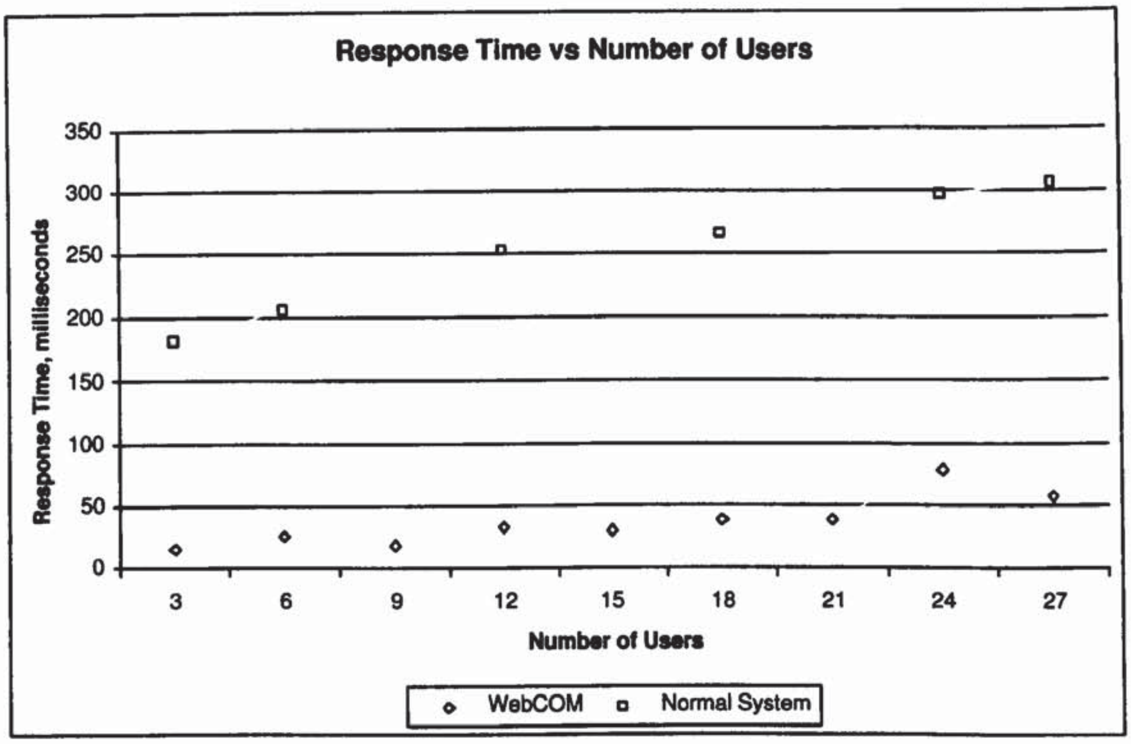


Figure 5.20: The Comparison of Response Time between WebCOM and Normal System vs Number of Users

5.8.1 Effect Number of Users

Figure 5.20 shows the response time for WebCOM and "normal system". It shows two lines indicated WebCOM and a "normal system". It is appear to be shows that WebCOM gives a quicker response to the users in a group responding in typically 15.5% of the average time required for a "normal system". Both of the systems generally showed slower response with increasing number of users. The rate of increase of response time is less for WebCOM than for the "normal" client/server system, suggesting that the improved response time under WebCOM will tend to become greater with larger numbers of users.

5.8.2 Effect Size of Packet

Figure 5.21 shows the comparison of average response time for WebCOM and the "normal system". It shows the effect of data size per packet on both systems. It

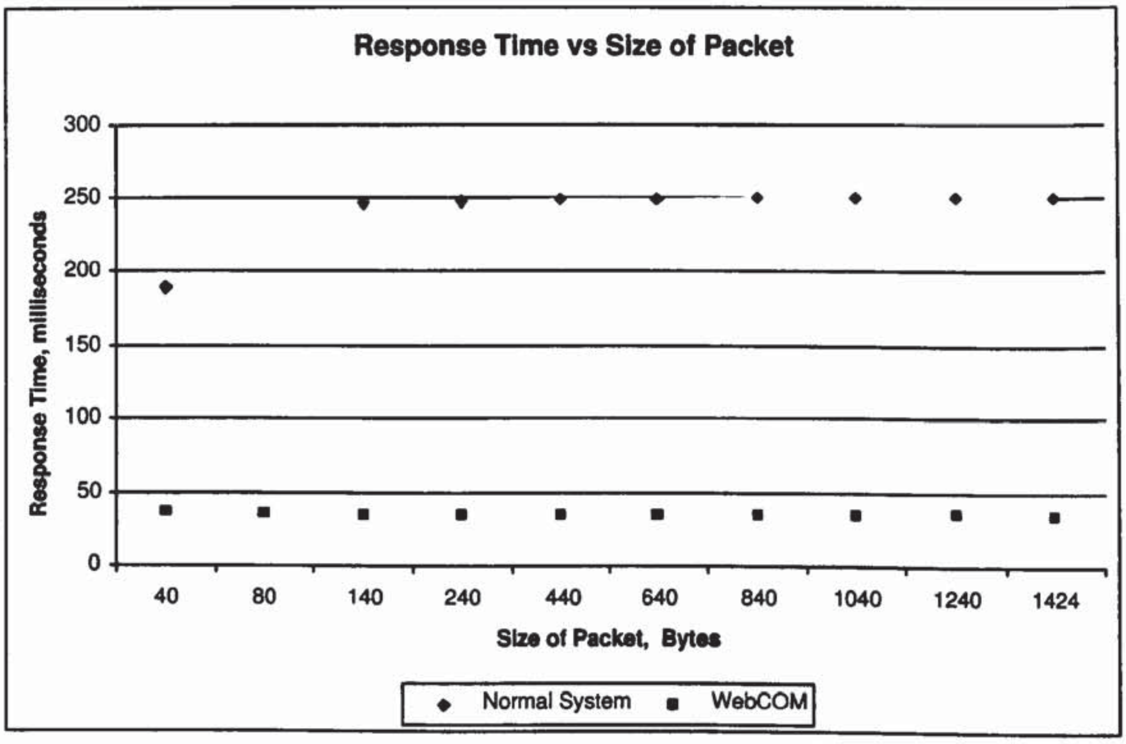


Figure 5.21: The Comparison of Response Time between WebCOM and Normal System vs Size of Packet

shows that WebCOM gives quicker response time compared with a "normal system" when carrying packets from 40 to 1424 bytes again requiring only 14.3% of the average time of the "normal system". One of the differences between WebCOM and "normal system" is that the "normal system" increases quickly at the beginning where data size per packet does not exceed 140 bytes and slowly increases approximately linearly afterward, while WebCOM shows a decay from the beginning. In conclusion, WebCOM appears to give better response time compared to the "normal system" as packet size is varied and its advantage will be maintained at larger packet size.

5.8.3 Throughput

Figure 5.22 shows the comparison between WebCOM and the "normal system" on throughput based on number of users. It shows that throughput of WebCOM falls with greater number of users, while throughput for the "normal system" remains generally constant.However even for the higher number of users, where the margin is smallest,

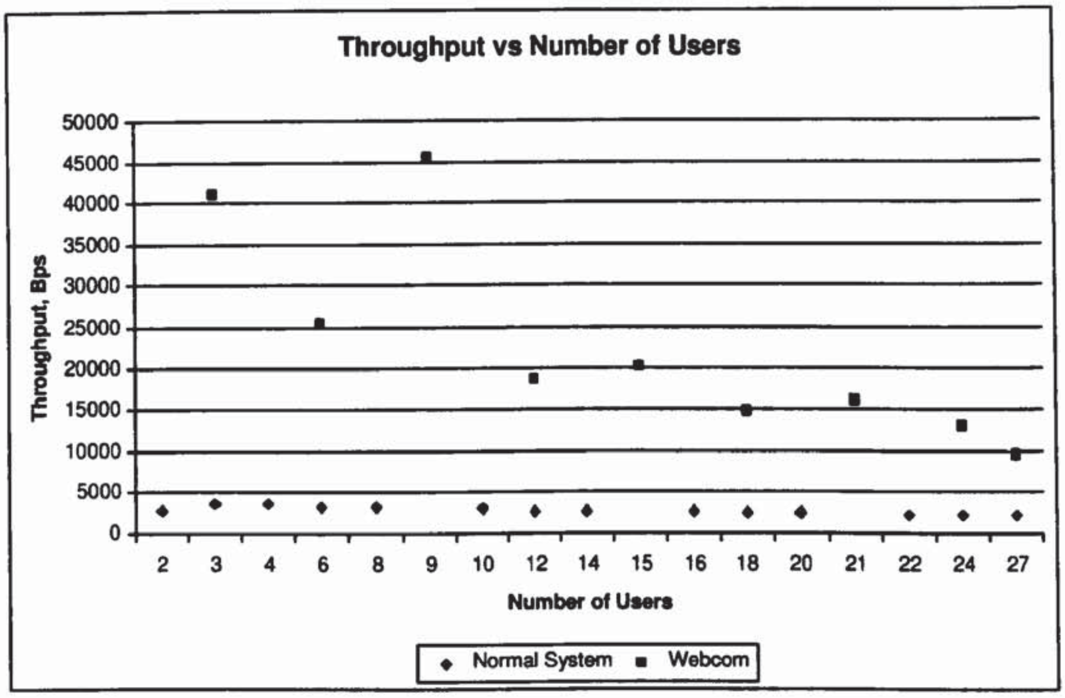


Figure 5.22: The Comparison of Throughput between WebCOM and Normal System vs Number of User

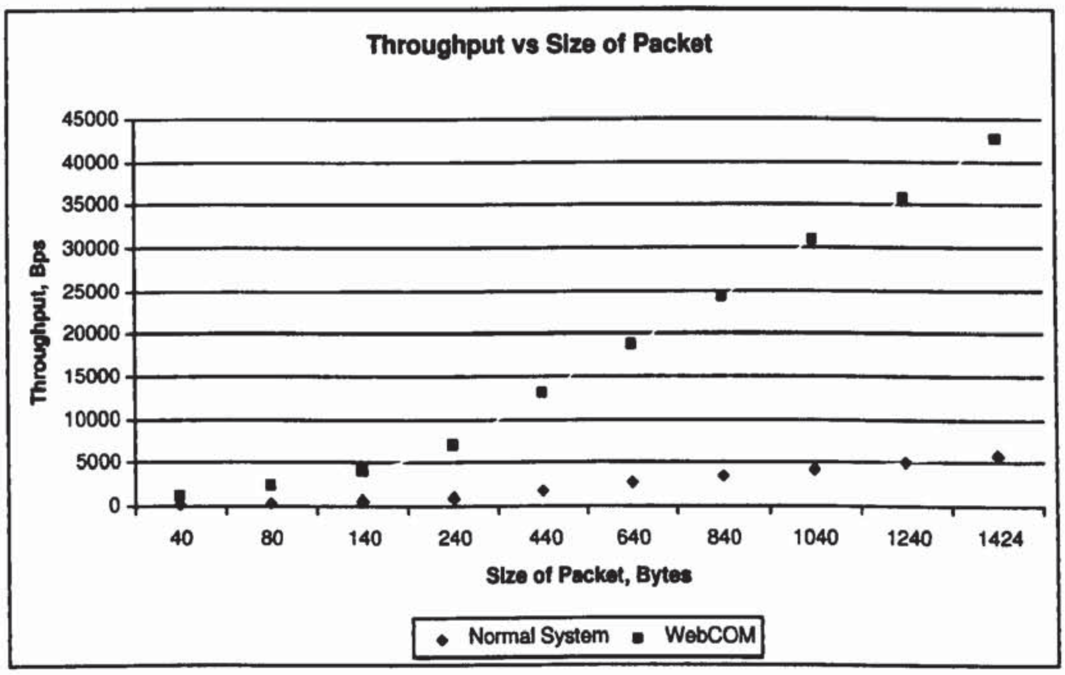


Figure 5.23: The Comparison of Throughput between WebCOM and Normal System vs Size of Packet

throughput of WebCOM is still 6 times that of a "normal system".

Figure 5.23 shows the comparison between WebCOM and "normal system" on throughput based on packet size. Here WebCOM performs much better than the "normal system", with the advantage growing with larger packet size. For small packets, WebCOM is 5 times better, but for large packets 7 times.

5.9 Result on Functionality Experiment

Table 5.1: The Comparison Between Prototype Run on WebCOM and Client/Server Unicasting

System / **Features**	**Prototype on WebCOM**	**Prototype on client/server unicasting**	**Supported by**
Scalability	Support many users with **high** network performance	Support many users with **low** network performance	Measured
Portability	Support major platforms: UNIX, Linux, Windows, Macintosh	Support major platforms: UNIX, Linux, Windows, Macintosh	Used in different platforms
Accessibility	Works even **without** server	Works only **with** server	Experiment

This section presents the results for the functionality of WebCOM System. Table 5.1 shows the result from the experimental evaluation where the system is implemented in order to evaluate the objectives of WebCOM as described before in Chapter 1, Section 1.3. In brief the objectives are summarised as follow:

- WebCOM is able to support many users with high network performance. This result was established by the experimental testing where the number of users was the independent variable.
- WebCOM is able to work on main platforms such as UNIX, Linux, Windows, and Macintosh. The well known client/server also supports this feature. This was established by running the system on different platforms.

- WebCOM is able to work in a peer-to-peer mode environment where no server exists. This was established by disabling the server. The server is however required for asynchronous type groupware application.

Chapter 6

Research Evaluation and Future Work

6.1 Introduction

This chapter discusses the evaluation of the thesis as a whole and points out several suggestions for future work. This chapter mentions again the objectives of work of this thesis to highlight the aim of the research. Then the evaluation of each stage in this thesis is provided. Lastly, suggestions are offered for continuing the project in future work.

6.2 Aim of Research

This section highlights the aims of this thesis again. The aims of the research were to address problems in the CSCW field which arise in implementation of CSCW applications. The problems are performance, scalability, accessibility, reliability and portability. In order to address these problems, the work in this thesis proposes WebCOM, an extension of communication frameworks by:

1. merging the client/server and peer-to-peer model as a hybrid architecture for WebCOM.

2. integrating a Reliable Multicast Protocol (RMP) for implementation of WebCOM to provide reliable data transmission using multicasting over network.

3. supporting groupware application tasks over the Internet to give quick response and to be reliable and scalable.

4. extending the scalability of existing technology to manage a large number of users in a group.

In addition, this research will:

1. develop a generic CSCW application that runs on the WebCOM framework. Then, a test will be done on the CSCW application to confirm it works under WebCOM.

2. test the performance of WebCOM in a collaborative environment. A comparison will be done by developing the same CSCW application running on a client/server model using unicasting.

Although this research has been developed to address issues of performance, portability, scalability, accessibility and reliability in groupware implementation over the Internet, the research has some limitations:

- WebCOM deals with small simultaneous users in a group and does not deal with high number of simultaneous users in one group.

- WebCOM tests only on basic text data and does not tested on streamed data.

- WebCOM deals with light-weight reliability and does not deal with high reliability data transfer.

- WebCOM deals with generic functions of CSCW applications and does not deal with complex functions of CSCW applications.

6.3 Evaluation of The Research

This section presents the evaluation of this research, which includes the literature review, the formulation of the research problem, the design, the prototype, the testing, and the analysis of this thesis.

6.3.1 Literature Survey

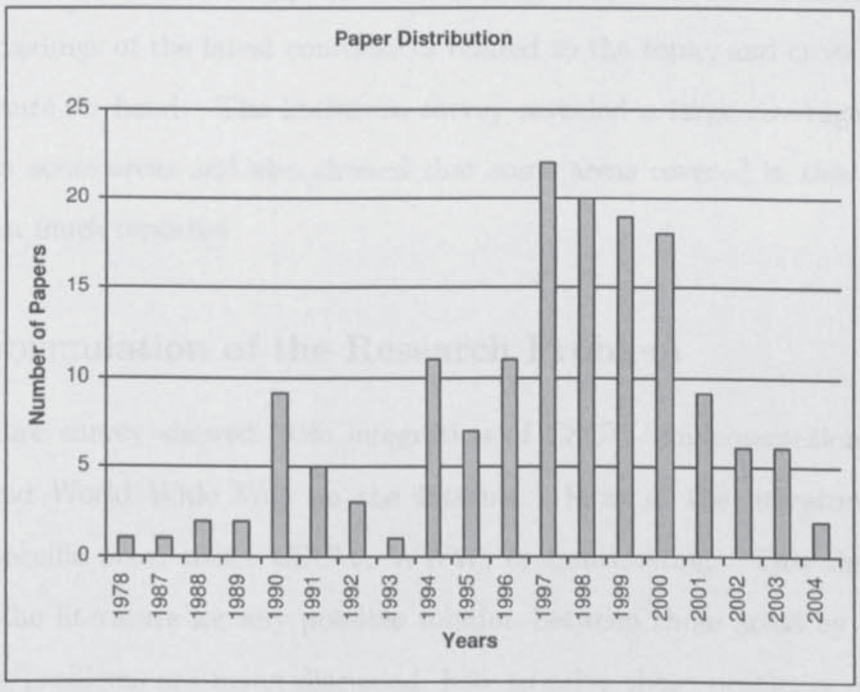


Figure 6.1: Distribution of Papers

An extensive survey of the literature was successfully completed. A large number of references (approximately 150 papers) were collected from various sources such as refereed journals, theses, books, conference publications, indexed search on electronic libraries, as well as sources on the Internet (refer Figure 6.1). The scope of literature found was spread evenly, covering the main issue of CSCW, and related communication technology. Three main areas were surveyed as literature i.e. CSCW, WWW and multicasting.

Many tools were used to gather all the literature, such as all the available searching tools and databases[1] in Aston University library, abstracts and index of theses, and Internet Explorer searching tools. The literature found was filtered according to the scope of this research. The literature in CSCW, World Wide Web and multicasting covered a specific issues of collaboration work, architecture and network communication.

The wide range of source of information ensured the survey found most of the related literature. This was supported by checking with the index of theses[2], citation index, proceedings of the latest conferences related to the topic, and cross referencing from literature on hand. The literature survey revealed a large coverage of current literature in some areas and also showed that some areas covered in this thesis have not yet been much reported.

6.3.2 Formulation of the Research Problem

The literature survey showed little integration of CSCW implementation with multicasting and World Wide Web on the Internet. Most of the literature discussed mainly a specific area, either CSCW, WWW, or multicasting. This thesis has investigated the literature for any possible relation between these areas by considering what issues/problems are being discussed, how to solve those problems, what methods/techniques being used, and the outcome/result. A big picture of problems, techniques, and results is produced in order to find an original approach in this area to investigate. A number of issues on CSCW implementation has been pointed out in this study. Most of the material collected on each of the three main topic is up-to-date (refer Figure 6.1).

The material from each of the three main topics provided a set of components that could be combined and extended to meet the aims of the thesis. This study

[1]BIDS(Bath Information and Data Services) is a collection of bibliographic databases.
INSPEC is a bibliographic database, providing references and summaries of mostly journal articles and conference papers.
IDEAL(International Digital Electronic Access Library) is an online electronic library
[2]`http://www.theses.com/`

used tables (refer Table 3.2) to allow clear identification of particular areas not yet covered and confirmed that the thesis could contain novel elements. The study has clearly differentiated this other work from other related works in the CSCW area and in methods of implementation.

There are other issues not discussed in this thesis for example, security and human interaction. The omission of consideration of other issues in CSCW such as security, human interface, and others did not affect the ability to develop, test and prove WebCOM.

6.3.3 The Framework

The needs of CSCW for groupware applications on the Internet led to a proposal for developing a framework to addresses the issues of accessibility, portability, scalability, reliability and performance for groupware applications. An extensive comparison of communication frameworks for CSCW applications in collaborative environments over the Internet has been made from related works.

The design of the WebCOM System in this thesis was carried out using UML although there are other tools that could be used for designing a system. However in this thesis, UML allowed the WebCOM System to be described fully and made it easy to carry out the development process, which used the Java language as an object oriented programming language for developing the system. In this thesis, the WebCOM System was designed using UML and this has led to successful deployment of WebCOM supporting CSCW applications on collaborative environment which was tested in the experiment phase.

The framework design takes into consideration technology availability. The framework design was based on the approach of JASMINE's design (El-Saddik *et al.*, 2000), which explained clearly the communication design using Java applets. The approach of using Java applets in JASMINE is suited to the WebCOM framework. However, the design detail of JASMINE is different from WebCOM in architecture, communication method, and working environment.

Basic tools of a generic CSCW application were considered for the prototype system design and development. The selection of CSCW tools was based on the objective of this thesis, which is that a prototype that can be used to evaluate the ability of the WebCOM System as a communication framework for CSCW applications working in a collaborative environment on the Internet. In order to test the ability of the framework, a selection of basic tools with simple functions that cover the different CSCW categories was chosen to test the overall ability of the framework to support CSCW applications.

The architecture used to implement the WebCOM System is hybrid. This architecture was selected based on the literature survey. The hybrid architecture is a combination of architectures to make communication of processes for certain systems more effective. This thesis has chosen client/server and peer-to-peer process as a combination for the hybrid architecture. The hybrid architecture proved able to support the needs of WebCOM System to work for collaborative Web environment without problems, and proved robust. The architecture gives the WebCOM System good performance, and allows WebCOM to work without a server. Further work might be implemented by choosing other architectures, but the core of the WebCOM framework does not need to change.

Multicasting was considered as the main method of communication in WebCOM System. From the literature survey, only few works in CSCW use multicasting in their implementation. Most CSCW uses unicasting. According to Maihöfer (2000), multicasting is considered as an effective means of communication that provides scalability and quick response in group communication. The use of multicasting in CSCW implementation is a novel part of this thesis.

LRMP was integrated in the WebCOM framework design in order to obtain reliability and to enable multicasting environment for WebCOM System. From testing, it was proved that by using LRMP in the WebCOM framework, the WebCOM System works without multicast enabled-routers. It also gives a higher percentage of reliability for data transmission over the network with the average for lost packets 5.76 % and average percentage for reliability on successful packets on the Internet 94.24% (refer Subsection 5.6.3). The LRMP APIs coded in the Java language simplified the networking coding for the WebCOM System. LRMP is easy to install on any platform. While implementing the WebCOM System with LRMP, no problem was detected and it proved robust. LRMP should work with a bigger system because LRMP only focuses on the network level, thus LRMP is suited to further expansion of this project.

6.3.4 Prototype and Implementation

The construction of the prototype was aimed at implementing the communication framework to meet the goals of the communication issues of CSCW for collaborative work on the Internet. The existing tools for CSCW applications were investigated and analysed. Based on this information, a set of tools was proposed for developing the prototype system. The criteria for selecting the tools was to provide basic functions for CSCW applications which use both asynchronous and synchronous modes.

The prototype was constructed based on the design phase. The prototype was developed using the Java language. The Java language was chosen because it offers the ability to work on any platform and also provides APIs to simplify the coding process. Although Java has a number of limitation/constraints such as slow downloading, security, and others, these limitations did not affect the objectives and implementation of the WebCOM System. The Java networking API (LRMP) greatly simplified coding of WebCOM.

World Wide Web environment was chosen as an environment for collaborative work on the Internet because the advantages provided by WWW, discussed in the literature survey, provide the environment required by the WebCOM System. From the tests, the WWW proved to support all the needs of WebCOM System, with no obvious performance difficulties. If further work is done on the WebCOM System, the WWW environment would be suitable to support the work.

Ethernet technology was used to implement WebCOM System because of the availability and low cost of this technology to the end-users. The Ethernet proved to support all needs of the WebCOM System with fast enough response time and high reliability. This technology is robust, easy to install and it is easy to run the WebCOM System with no complex configuration and increase of maintenance required. If further work is done, this technology remains preferable to other technologies to support WebCOM.

In order to implement the WebCOM System on the Ethernet technology, the system was implemented on a Computer Science laboratory, Aston University which has an open network to the Internet. The laboratory was chosen as a testbed for this study because the environment, the number of machines, location and configurations of the laboratory meets the minimum requirement of WebCOM System. From the testing done, it was proved that the chosen testbed gives good performance and reliability with WebCOM. It also provides an environment in which to test WebCOM and the "normal system" without any problems and not require complex configuration in order to distinguish between these systems. The testbed also works for testing of availability, portability and scalability for WebCOM System.

Although the WebCOM System was successfully designed, developed, and tested, its performance in comparison to other frameworks, apart from client/server, was not tested. Further work might evaluate WebCOM compared with other frameworks in order to identify their relative strengths and limitations. WebCOM could be tested on other technologies such as ATM and SONET to evaluate the efficiency of WebCOM.

6.3.5 Experimental Testing and Analysis

There were two series of tests in this thesis. The first series were tests on the WebCOM System and the second series was on a "normal system" that works on client/server unicasting. The objective of the testing was to evaluate WebCOM in term of performance metrics (response time, throughput) and QoS metrics (packet loss) for WebCOM to evaluate its reliability. The second series of testing was used to compare the performance metrics with the "normal system" system in order to distinguish the performance efficiency between WebCOM and the "normal system" which tested on the same testbed.

To ensure the comparison can be made between WebCOM System and client/server:

- the same testbed was used
- the same environment was used
- the same data was used
- the same number of recipient was used.

These features were chosen to get a fair comparison between these system.

The experiment used two ways of taking results. All PCs were numbered according to their location. The first method, the group was formed by sequential numbering (see Figure 5.2). The results were unpredictable because of the effect of the switches. The second method formed the group according to the location of a PC on different switches. The result more predictable than the first method. Formation of the group by random selection was not done, and could form the basis of further work.

Three issues are considered while experimenting the system: clock synchronisation, reading accuracy, and network traffic. All of them are mentioned in Chapter 1 and 5 and discussed in detail in Appendix G. By considering these issues, the outcomes from the experiment are considered accurate. The accuracy of the experimental result is in milliseconds.

The results from the experiment were used in performance analysis, which determined parameters influencing the response time, throughput, and packet loss. These parameters were used in the evaluation of the WebCOM System, based on the literature review where most of the metrics used in this thesis were also used in other work. The selected metrics allow clear distinction between the WebCOM System and the "normal system". These metrics allowed the overall conclusion to be drawn on reliability and performance. By using metrics that other researchers have used, we are measuring aspects of performance considered important by other researchers.

It has been shown in this thesis that WebCOM gives a faster response and is more scalable compared with a "normal system", portable so it can execute on any platform, and accessible to the system even when the Web server is down.

6.3.6 Generalisation from this Work

The architecture of the test framework was designed with the CSCW application as a separate module from the communication modules. This architecture would be expected to allow substitution of another application in place of the CSCW application without interference with the network performance measurements that were used in this work.

For this reason it is probable that the conclusions on the improved performance of WebCOM would extend to other distributed applications.

6.3.7 Overall Critique

In the development of the WebCOM System, the basis for development of the scalable framework is review of the literature of existing practices. In an ideal situation, the development of the framework should have more data in the form of observation of a real working environment on top of review of existing practices. In the author's judgement the lack of observation may influence the detail of WebCOM, but the principle derived from the literature is well formed.

The implementation of the prototype only concentrated on a small aspect of collaborative work tasks. However these were carefully chosen to represent different issues in CSCW which increases confidence that the work is representative of CSCW system applications. The framework developed could be used with confidence as a general collaborative model that supports other tasks.

The evaluation in this research is only on the response time, throughput, and packet loss. No study has been conducted on other performance metrics and QoS metrics in order to determine of effectiveness of the collaborative processes on the quality of the application produced. These were considered less important than the chosen metrics in accessing accessibility, portability, scalability, reliability and performance. Further testing is needed in order to determine the effectiveness of WebCOM System under these other metrics.

A more comprehensive experiment, for example, an experiment in a wide area network connected with more routers, could be conducted. This would allow more confidence in the statistical conclusion, for example, the analysis of response time and throughput. Ideally, the testing should be conducted in a *"real-life"* working environment.

The implementation of the prototype on the Web using Java does have it strengths and its weaknesses. One of the problems is because of the interpreted nature of Java and the reliance on the World Wide Web infrastructure. This combination made the system slow in terms of loading time and response to user requests from the Web server. However, the functional and operational performance of the WebCOM System does not effect the loading or request time of an applet because once the WebCOM System is downloaded, it works independently. Although there exist issues of different versions of Java releases which might show an inconsistency in the features supported, the core of WebCOM framework is built from the basic networking features of the Java API which are less likely to change.

Last but not least, the trial system lacks security features. Although there is some rudimentary access control built into the system, there exist a number of other security questions. Ideally, the WebCOM architecture would be constructed in an Internet environment in which the WebCOM is implemented behind a firewall for protection against unauthorised access from the Internet, but in the implementation, no such firewall exists. The free-flowing nature of the information across the internet also creates another security issue. The information needs to be protected from unauthorised access.

6.4 Recommendations for future research

The research has opened up to a number of possibilities for future work. The suggested list is provided below:

- There are many ways to increase network performance. The work in this thesis, concentrated on implementing WebCOM in Ethernet. Other work can be done using other methods such as ATM and SOP.
- The work described in this thesis excluded security issues. It is proposed that further work should investigate these aspects. Furthermore, security in multicasting is an area where work is not yet established. Security features might be implemented in the WebCOM System.
- This thesis has developed a communication framework and evaluated it without it being tested on the end-user. The relation between human factors and the efficiency of the framework could be investigated. This may result in finding a more suitable method for presenting the application to users on the Internet.
- The performance of WebCOM might vary when different protocols are used. Issues of protocols implemented on WebCOM should be investigated to establish the best protocol.
- Implementation of different technologies might give some advantages such as implementation on a wireless communication testbed, or expanding the platform base of WebCOM to include handheld computing devices.
- The hybrid architecture might be expanded to adapt to other architectures such as replicated and distributed architectures.

- Cache or proxy server might be used to increase the availability and performance of the Web server services.

- Different channels might be implemented to enhance the capabilities of multicasting communication in WebCOM. The channels might be used to carry controls and data.

Chapter 7

Conclusions

7.1 Conclusions

CSCW is a field that studies the use of computer technologies to enable people to work together by cooperating directly using shared resources in a small and closed group (Coulouris, 2001). The cooperation among users is normally enhanced by tools which support the users in accomplishing their goals. However, these tools have sometimes failed to serve the needs of users and the tools have been abandoned by users. The literature suggests that many of these problems can be addressed by enhancing accessibility, portability, scalability, reliability and performance in CSCW systems. The work in this thesis proposes a communication framework for CSCW systems to address these issues.

This study is differentiated from other related works by the implementation of a hybrid architecture (client/server and peer-to-peer), integrating a different reliable multicasting protocol which offers many benefits for this study, LRMP, working with many users (more than two), working with two modes of communication (asynchronous and synchronous), and using multicasting as a main method of communication among users in a group.

Three fields were investigated in seeking solutions: CSCW as a case study for the research, World Wide Web and Java languages as the environment that provide acces-

Table 7.1: Conclusion Based on Objectives of This Thesis

Objectives	How achieved	Result	Comment
Architecture	Merging the client/server and peer-to-peer architecture. Achieved by allowing communication between server-to-client, client-to-server and client-to-client.	Client and server able to communicate with each other on different architectures.	
Protocol	LRMP integrated into the multicasting communication in WebCOM to enable it to work although there is no mrouter in network.	LRMP works on multicasting as a component in WebCOM framework.	
Quick response	Implementing multicasting in CSCW applications. Experiments have been done to evaluate the performance of WebCOM. A comparison has been made by implementing unicasting in CSCW application and testing on a same testbed.	The performance has been tested and WebCOM was faster compared to a unicasting client/server.	
Scalable	Runs a prototype that uses WebCOM in a testbed with a number of participants.	Gives good scalability and performance to support a larger number of users in a group.	The experiment is done with 27 machines only. A simulation test can be done to see the scalability of WebCOM.
Reliable	Implement reliable multicast protocol, LRMP, into WebCOM framework. An experiment has been done to evaluate the reliability of LRMP by measuring packet loss as QoS metric.	The experiment result gives some percentage of packet loss on WebCOM.	
Manage large user	Implement the prototype in a large number group.	A large number of users in a group can be managed easily.	
Prototype	Develop a prototype that functions as a generic CSCW application which offers basic functions of CSCW application. The prototype is implemented on the WebCOM and on the client/server unicasting as a normal system.	All functions in the prototype are functioning properly.	
Testing	Experiment is done to evaluate the WebCOM system and make a comparison with a normal system. The experiments are done on a testbed.	The evaluation through experiment gives a fair comparison for WebCOM and a normal system because use same testbed.	

sibility and portability; and multicasting as the communication method that provides reliability, scalability and performance. From a combination of these three different research area (CSCW, WWW, and multicasting), this study has enhanced existing Web-based communications by proposing a communication framework called Web-COM.

In this thesis, WebCOM has been built as a system that consists of WebCOM Communication Module, WebCOM Client, and WebCOM Server; which form the WebCOM System. WebCOM System was implemented using the Java language and Java APIs. The main transmission method used in WebCOM System is multicasting. WebCOM System was designed to work on a hybrid architecture, a combination of peer-to-peer and client/server architecture, which gives the advantages of both peer-to-peer and client/server architectures. A prototype of some CSCW applications was developed to be implemented on WebCOM System. The prototype was implemented on WebCOM System to confirm the functions of the communication framework in a CSCW context.

The framework has been tested for performance by measuring the response time and throughput. The same prototype was run on a unicasting client/server model to make comparison on specified features with the WebCOM System. The features considered in this thesis are the effect of user number and packet size on response time and throughput. The WebCOM System has also evaluated one QoS metric, i.e. packet loss to measure the reliability of WebCOM System on transferring data over network. Other issues on scalability, accessibility and portability were tested to evaluate the ability of WebCOM System serving CSCW application on the Internet. The prototypes that runs on both system were implemented and tested in the same testbed constructed in a Computer Science laboratory.

Analysis of experimental results show that WebCOM gives faster response time and higher throughput compared to the "normal system" on unicasting client/server architecture within the research scopes. The overall reliability of WebCOM System was 94% as measured by packet loss. The WebCOM System is able to work on any platform, and function when a server breaks down. The scalability of WebCOM is good in terms of supporting a small number of participants in a group (which imply to CSCW area), with faster response time compared to the "normal system". However, the scalability has not been tested on a large number of users and on real implementation on the Internet, which may measure more clearly the ability of multicasting.

In summary, WebCOM serves as a communication framework for CSCW applications, offering faster response than client/server unicasting systems. Accessibility is improved by using multicasting with the advantage of peer-to-peer architecture, portability is achieved by using the Java language to develop both communication framework and prototype, reliability is improved by integrating a reliable multicasting protocol into WebCOM, and performance is improve by implementing multicasting as the communication within a small user group. The use of modular architecture in the test framework makes it likely that WebCOM would offer improved performance in a wider range of distributed applications than CSCW.

The work on WebCOM has been reported in peer-reviewed conferences. The papers are:

- Serverless Web-Multicast Chat System for Multi-users at International Conference on Web Engineering, Spain (Ngadi and Doherty, 2003b).
- Multi-users Shared Whiteboard under Web- multicasting in International Symposium on Information and Communication Technologies, Dublin (Ngadi and Doherty, 2003a).
- Web-multicast Chat System for Collaborative Learning in International Conference on e-Society, Portugal (Ngadi and Doherty, 2003c).

Overall the research has achieved its aim, providing an improved framework for groupware applications on the Internet. WebCOM provides a good framework CSCW in terms of performance, accessibility, and portability. For future work, WebCOM could be used for other types of CSCW applications such as audio/video conferencing, tele-conferencing, and games applications. WebCOM could also be used for other disciplines such as teaching, meetings, or discussion, where a group environment is appropriate.

Bibliography

Adoud, H., Rondeau, E., and Divoux, T. (2001). Configuration Of Communication Networks By Analysing Co-Operation Graphs. *Computer Communications*, **24**(15-16), 1568–1577.

Aggarwal, S., Paul, S., Massey, D., and Caldararu, D. (2000). A Flexible Multicast Routing Protocol For Group Communication. *Computer Networks*, **32**(1), 35–60.

Agostini, A., De Michelis, G., Divitini, M., Grasso, M., and Snowdon, D. (2002). Design and Deployment Of Community Systems: Reflections On The Campiello Experience. *Interacting with Computers*, **14**(6), 689–712.

Ahmed, M., Krishnamurthy, S., Katz, R., and Dao, S. (2002). An Architecture For Providing Range Extension By Deploying Mobile Gateways in Ad Hoc Network. Available at `http://citeseer.nj.nec.com/551324.html`.

Al-Shaer, E. and Tang, Y. (2001). Toward Integrating IP Multicasting In Internet Network Management Protocols. *Computer Communication*, **24**(5-6), 473–485.

Albanna, Z., Almeroth, K., Meyer, D., and Schipper, M. (2001). IANA Guidelines for IPv4 Multicast Address Assignments. RFC 3171. IETF.

ANSI/IEEE (1991). Standard Glossary of Software Engineering Terminology. Technical Report STD-729-1991, ANSI/IEEE.

Arlitt, M., Friedrich, R., and Jin, T. (2000). Performance Evaluation Of Web Proxy Cache Replacement Policies. *Performance Evaluation*, **39**(1-4), 149–164.

Ballardie, A. (1995). Core Based Tree (CBT) Multicast - Architectural Overview and Specification. Available at `http://citeseer.nj.nec.com/ballardie95core.html`.

Balter, R., Atallah, S., and Kanawati, R. (1995). Architecture for Synchronous Groupware Application Development. In *Proceedings of the Sixth International Conference on Human-Computer Interactions (HCI International '95*, pages 371–379, Tokyo, Japan. Elsevier.

Barford, P. and Crovella, M. (1999). Measuring Web Performance in the Wide Area. Technical Report 1999-004, Computer Science Department, Boston University.

Basile, C., Wang, L., Kalbarczyk, Z., and Iyer, R. (2003). Group Communication Protocols Under Errors. In *Proceeding of 22nd International Symposium on Reliable Distributed Systems*, pages 35–44, Florence, Italy. IEEE Computer Society.

Beca, L. and et. al. (1997). TANGO INTERACTIVE - A Collaborative Environment for the World-Wide Web. Technical report, Northeast Parallel Architectures Center, Syracuse University, Syracuse, New York. Available at `http://www.npac.syr.edu/`.

Benbunan-Fich, R., Hiltz, S. R., and Turoff, M. (2003). A Comparative Content Analysis Of Face-To-Face Vs. Asynchronous Group Decision Making. *Decision Support Systems*, **34**(4), 457–469.

Bentley, R., Horstmann, T. C., and Trevor, J. (1997). The World Wide Web as Enabling Technology for CSCW: The Case of BSCW. *Computer Supported Cooperative Work*, **6**(2-3), 111–134.

Berners-Lee, T., Cailliau, R., Luotnen, A., Frystyck Nielsen, H., and Secret, A. (1994). The World Wide Web. *Communications of the ACM*, **37**(8), 76–82.

Borella, M., Swider, D., Uludag, S., and Brewster, G. (1998). Internet Packet Loss: Measurement and Implications for End-to-End QoS. In *Proceedings of International Conference on Parallel Processing*. Available at `http://citeseer.nj.nec.com/borella98internet.html`.

Bouras, C., Destounis, P., Garofalakis, J., Triantafillou, V., Tzimas, G., and Zarafidis, P. (1999). A Co-Operative Environment For Local Government: An Internet-Intranet Approach. *Telematics and Informatics*, **16**(1-2), 75–89.

Bouras, C., Destounis, P., Garofalakis, J., Gkamas, A., Sakalis, G., Sakkopoulos, E., Tsaknakis, J., and Tsiatsos, T. (2000). Efficient Web-Based Open And Distance Learning Services. *Telematics and Informatics*, **17**(3), 213–237.

Bourges-Waldegg, P. and Scrivener, S. A. R. (1998). Meaning, The Central Issue In Cross-Cultural HCI Design. *Interacting with Computers*, **9**(3), 287–309.

Broll, W., Meier, E., and Schardt, T. (2000). The Virtual Round Table - A Collaborative Augmented Multi-User Environment. In *Proceedings of the third international conference on Collaborative virtual environments*, pages 39–45. ACM Press.

Bullen, C. and Bennett, J. (1990). Learning from User Experience with Groupware. In *Proceedings of ACM CSCW'90 Conference on Computer-Supported Cooperative Work*, volume 1, pages 291–302, New York. ACM.

Burns, E. (1995). Webcast - Collaborative Document Sharing via the MBONE. Available on-line. `http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/CCI/webcast.html`.

Butler, T. and Coleman, D. (2003). Models of Collaboration. Collaborative Strategies LLC. `http://www.collaborate.com/publication/newsletter/publications_newsletter_september03.html`.

Cain, B., Deering, S., and Thyagarajan, A. (2002). Internet Group Management Protocol Version 3. RFC 3376, IETF.

Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., and Seguin, C. (1998). Java Object-Sharing in Habanero. *Communications of the ACM*, **41**(6), 69–76.

Chaun, T. K. (1999). Small group CSCW using Web resources: A Case Study. In *Proceedings of 8th International Conference on Human Computer Interaction and*

Special Session on Intelligent Tutoring and Learning Environments, pages 788–92, Mahwah, NJ, USA. Lawrence Erlbaum Associates.

Chawathe, Y., Fink, S. A., McCanne, S., and Brewer, E. A. (1998). A Proxy Architecture For Reliable Multicast In Heterogeneous Environments. In *Proceedings of the sixth ACM international conference on Multimedia*, pages 151–159. ACM Press.

Cheung, S. C. and Chanson, S. T. (1999). A Model-Based Authorware For The Construction Of Distributed Multimedia Systems. *Information and Software Technology*, **41**(11-12), 175–727.

Clarence, A. and Wainer, J. (1994). Goal Based Models of Collaboration. *Collaborative Computing*, **1**(1), 61–86.

Condon, C. (1993). *The Computer Won't Let Me: Cooperation, Conflict and Ownership of Information*, chapter CSCW: Cooperation or Conflict?, pages 171–185. Springer-Verlag, London.

Coulouris, G., Dollimore, J., and Kindberg, T. (2001). *Distributed System: Concepts And Design*. Addison-Wesley, 3rd edition.

Coulouris, G. F. (2001). *Distributed Systems: Concepts and Design*. Addison-Wesley, Harlow, 3rd edition.

Crowley, T., Milazzo, P., Baker, E., Forsdick, H., and Tomlinson, R. (1990). MMConf: An Infrastructure For Building Multimedia Applications. In *Proceeding of CSCW 90*, pages 329–342, Los Angeles, CA.

Dauphin, G. (1996). mMosaic: Yet Another Tool Bringing Multicast to the Web. In *Proceedings of the Workshop on Real Time Multimedia and the WWW (RTMW '96)*, INRIA Sophia-Antipolis, France. W3C - World Wide Web Consortium.

Deering, S. (1989). Host Extensions For IP Multicasting. RFC 1112, IETF.

Deering, S. (1990). Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transaction on Computer Systems*, **8**(2), 85–110.

Deering, S. (1991). *Multicast Routing in a Datagram Internetwork.* Phd thesis, Stanford University.

Deering, S., Partridge, C., and Waitzman, D. (1988). Distance Vector Multicast Routing Protocol. RFC 1075, IETF.

Deering, S., Estrin, D., Fariancci, D., Handley, M., Helmy, A., Jacob son, V., Liu, C., Sharma, P., Thaler, D., and Wei, L. (1997). Protocol Independent Multicast-Sparse Mode (PIM-SM): Motivation and Architecture. Internet Draft draft-ietf-idmr-pim-arch-01.ps, IETF.

Desanctis, G. and Gallupe, R. B. (1987). A Foundation For The Study Of Group Decision Support Systems. *Management Science*, **33**(5), 589–609.

Dietz, M. A., Ellis, C. S., and Starmer, C. F. (1995). Clock Instability and its Effect on Time Intervals in Performance Studies. Technical Report CS–1995–13, Department of Computer Science, Duke University Durham, Durham, North Carolina.

Dix, A. (1996). Challenges and Perspective for Cooperative Work On the Web. In *Proceeding of ERCIM workshop on CSCW and the Web*, pages 143–157, Sankt Augustin, Germany.

Dix, A., Ramduny, D., and Wilkinson, J. (1998). Interaction In The Large. *Interacting with Computers*, **11**(1), 9–32.

Dörries, G. and Zier, L. (2001). How To Do High-Speed Multicast Right! *Computer Networks*, **37**(6), 717–728.

Dressler, F. (2002). An Approach for QoS Measurements in IP Multicast Networks, MQM - Multicast Quality Monitor. In *Proceedings of Third International Network Conference (INC 2002).* Available at `http://bsd.rrze.uni-erlangen.de/~fd/publications/inc2002-abstract_en.html`.

El-Saddik, A., Shirmohammadi, S., Georganas, N. D., and Steinmetz, R. (2000). JASMINE: Java Application Sharing in Multiuser INteractive Environments. In *Inter-*

active Distributed Multimedia Systems and Telecommunication Services, pages 214–226.

Ellis, C. A., Gibbs, S. J., and Rein, G. L. (1991). Groupware: Some Issues and Experiences. *Communications of the ACM*, **34**(1), 38–58.

Encarnação, J., Mengel, M., Bono, P., Bhm, K., Borgmeier, E., Brisson-Lopes, J., Hornung, C., Knierriem-Jasnoch, A., Koch, E., and Krömer, D. (1998). A Concept And System Architecture For IT-based Lifelong Learning. *Computers & Graphics*, **22**(2-3), 319–393.

Erikson, H. (1994). MBONE: The Multicast Backbone. *Communications of the ACM*, **37**(8), 54–60.

Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., P., S., and Wei, L. (1997). Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2117, IETF.

Fenner, W. (1997). Internet Group Management Protocol - Version 2. RFC 2236, IETF.

Fielding, R. and Kaiser, G. (1997). The Apache HTTP Server Project. *IEEE Internet Computing*, **1**(4), 88–90.

Floyd, S., Jacobson, V., Liu, C., McCanne, S., , and Zhang, L. (1997). Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, **5**(6), 784–803.

Foley, S. and Jacob, J. (1995). Specifying Security For CSCW Systems. In *Proceedings Computer Security Foundations Workshop*, pages 136–145, Kenmare, Co. Kerry, Ireland. IEEE Computer Society.

Fuentes, L. and Troya, J. (1999). A Java Framework for Web-Based Multimedia and Collaborative Applications. *IEEE Internet Computing*, **3**(2), 55–64.

Gall, U. and F.J., H. (1997). Promondia: A Java-Based Framework for Real-time Group Communication in the Web. Technical Report TR-I4-96-08, Friedrich-Alexander-University, Erlangen-Nrnberg, Germany.

Gemmell, J., Schooler, E., and Kermode, R. (1998). An Architecture for Mutlicasting Telepresentations. *Journal of Computing and Information Technology*, **6**(3), 255–272.

Geyer, W. and Weis, R. (2000). The Design And The Security Concept Of A Collaborative Whiteboard. *Computer Communications*, **23**(3), 233–241.

Gmez, E. J., Caballero, P. J., Malpica, N., and del Pozo, F. (2001). Optimisation And Evaluation Of An Asynchronous Transfer Mode Teleradiology Co-Operative System: The Experience Of The EMERALD And The BONAPARTE Projects. *Computer Methods and Programs in Biomedicine*, **64**(3), 201–214.

Greenberg, S. and Roseman, M. (1996). GroupWeb : A Groupware Web Browser. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, page 7. ACM Press.

Grudin, J. (1989). Why Groupware Applications Fail: Problems In Design And Evaluation. *Office, Technology and People*, **4**(3), 245–264.

Grudin, J. (1994). Computer Supported Cooperative Work: History and Focus. *IEEE Computer*, **27**(5), 19–26.

Hall, R. W., Mathur, A., Jahanian, F., Prakahs, A., and Rasmussen, C. (1996). Corona: A Communication Service for Scalable, Reliable Group Collaboration Systems (Preliminary Design). Technical Report CSE-TR-290-96, The University of Michigan.

Hallin, Z., Jinrong, Z., and Yongzhou, Z. (1999). Multimedia Conferencing System And Multicasting. *Tsinghua Science & Technology*, **4**(2), 1407–1411.

Handley, M. (1995). SDR: Session Directory Tool. University College London, Available at `ftp://cs.ucl.ac.uk/mice/sdr/`.

Handley, M. (1997). Multicast Address Allocation Protocol (AAP). Internet Draft draft-draft-handley-aap-00, IETF.

Handley, M. and Jacobson, V. (1998). SDP: Session Description Protocol. RFC 2327, IETF.

Handley, M., Thaler, D., and Kermode, R. (2000a). Multicast-Scope Zone Announcement Protocol (MZAP). RFC 2776. IETF.

Handley, M., Perkins, C., and Whelan, E. (2000b). Session Announcement Protocol. RFC 2974, IETF.

Handley, M., Jacobson, V., and Perkins, C. (2002). SDP: Session Description Protocol. Internet Draft draft-ietf-mmusic-sdp-new-10, IETF.

Hardjono, T. and Cain, B. (1999). A Secure Group Membership Verification Protocol for IP Multicast. In *Proceedings of the The Fourth IEEE Symposium on Computers and Communications*, pages 9–15, Red Sea, Egypt. IEEE.

Hermanns, O. and Engbrocks, A. (1994). Design, Implementation and Evaluation of a Distributed File Service for Collaborative Engineering Environments. In *Proc. of 3rd. IEEE Workshop on Enabling Technologies: Infrastructures for Collaborative*, pages 170–177, Morgantown, West Virginia. Technical University of Aachen, Computer Science Department, IEEE.

Hill, R. D. and Brinck, T. (1994). Groupware For Realtime Collaboration. In *Conference Companion On Human Factors In Computing Systems*, pages 369–370. ACM Press.

Holbrook, H. W., Singhal, S. K., and Cheriton, D. R. (1995). Log-based Receiver-Reliable Multicast For Distributed Interactive Simulation. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 328–341. ACM Press.

Hong, X., Gerla, M., Pei, G., and Chiang, C. (1999). A Group Mobility Model for Ad Hoc Wireless Networks. In *Proceedings of ACM/IEEE MSWiM'99*, pages 53–60, Seattle, WA.

Hummel, T., Schoder, D., and Strauss (1996). Why CSCW applications Fail: Problem in the Support of Lateral Cooperation and the Appropriateness of CSCW Applications. In *Proceedings of the Annual Meeting of the Northest Decision Science Institute (NEDSI)*, pages 1–20, St. Croix, USA. University Freiburg, Denmark.

Ingvaldsen, T., Klovning, E., and Wilkins, M. (2000). Determining The Causes Of End-To-End Delay In CSCW Applications. *Computer Communications*, **23**(3), 219–232.

Jasnoch, A. K. (2001). An Approach To Classify It-Based Teaching And Learning Environments. *Computers & Graphics*, **25**(5), 899–907.

John Riedl, M. S., J., H. S., and Mashayekhi, V. (1997). A Case Study of Distributed, Asynchronous Software Inspection. In *Proceedings of the 1997 International Conference on Software Engineering*, pages 107–117, Boston, USA.

Kandansky, M., Chiu, D., Wesley, J., and Provino, J. (2000). Tree-based reliable multicast (TRAM). Internet Draft, IETF.

Karger, D., Sherman, A., Berkheimer, A., Bogstad, B., Dhanidina, R., Iwamoto, K., Kim, B., Matkins, L., and Yerushalmi, Y. (1999). Web Caching With Consistent Hashing. *Computer Networks*, **31**(11-16), 1203–1213.

Keshav, S. and Paul, S. (1998). Centralized Multicast. Technical Report TR98-1688, Computer Science Department, Cornell University.

Klovning, E. and Bonaventure, O. (1996). Performance evaluation of the CSCW application JVTOS. In *Proceedings of 2nd COST Workshop on Teleservices and Multimedia Communications*, pages 78–97. Springer.

Kristoffersen, S. and Ljungberg, F. (1998). MobiCom: Networking Dispersed Groups. *Interacting with Computers*, **10**(1), 45–65.

Kumar, S., Radoslavov, P., Thaler, D., Alaettino, C., Estrin, D., and Handley, M. (1998). The MASC/BGMP Architecture For Inter-Domain Multicast Routing. *SIGCOMM Comput. Commun. Rev.*, **28**(4), 93–104.

Labonté, C. and Srinivas, S. (2000). Group Management Strategies for Secure Multicasting on Active Virtual Private Network. In *25th Annual IEEE Conference On Local Computer Networks (LCN'00)*, pages 213–222, Tampa, Florida. IEEE Press.

Lamport, L. (1978). Time, Clocks, And The Ordering Of Events In A Distributed System. *Communications of the ACM*, **21**(7), 558–565.

Lauwers, J. C., Joseph, T. A., Lantz, K. A., and Romanow, A. L. (1990). Replicated Architectures For Shared Window Systems: A Critique. In *Proceedings of the Conference on Office Information Systems*, pages 249–260. ACM Press.

Lee, M. J., Han, C. Y., Ahn, G. T., Kim, J. H., Moon, N. D., and Jung, M. H. (2000). CoWare: A Web-Based Groupware For Effective Collaboration. In *Proceedings of the 4th Korea-Russia Int. Symp. on Science and Technology*, volume 3, pages 128–133, Piscataway, NJ, USA.

Leung, P. and Cheung, S. C. (1999). A CSCW Framework For The Flexible Coupling Of Groupware Widgets. In *Proceeding of the Fifth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'99)*, pages 9–20, Los Alamitos, CA, USA. IEEE Computer Society.

Levine, B. N. (1996). *A Comparison of Known Classes of Reliable Multicast Protocols.* Master thesis, University of California, Santa Cruz, Santa Cruz, California.

Liao, T. (1997). WebCanal: A Multicast Web Application. *Computer Network and ISDN Systems*, **29**(8-13), 1091–1102.

Liao, T. (1998). Light-weight Reliable Multicast Protocol. Technical report, INRIA.

Liao, T. (2003). Light-weight Reliable Multicast Protocol as an Extension to RTP. Available at `http://webcanal.inria.fr/lrmp/lrmp_rtp.html`.

Lin, J. and Chang, R. S. (1999). A Comparison Of The Internet Multicast Routing Protocols. *Computer Communications*, **22**(2), 144–155.

Lin, Q., Low, C., Ng, J., Bu, J., and Liu, X. (2003). Multiuser Collaborative Work In Virtual Environment Based CASE Tool. *Information and Software Technology*, **45**(5), 253–267.

Liu, Z., Niclausse, N., and Jalpa-Villanueva, C. (2001). Traffic Model And Performance Evaluation Of Web Servers. *Performance Evaluation*, **46**(2-3), 77–100.

Lynne Markus, M. and Connolly, T. (1990). Why CSCW Applications Fail: Problems In The Adoption Of Interdependent Work Tools. In *Proceedings of the 1990 ACM Conference on Computer Supported Cooperative Work*, pages 371–380. ACM Press.

Maihöfer, C. (2000). A Bandwidth Analysis of Reliable Multicast Transport Protocols. In *Proceedings of the NGC 2000 on Networked Group Communication*, pages 15–26, Palo Alto, CA, USA. ACM.

Marsic, I. (1999). DISCIPLE: A Framework For Multimodal Collaboration In Heterogeneous Environments. *ACM Comput. Surv.*, **31**(2es), 4.

Matta, I. and Eltoweissy, M. (1997). A Scalable QoS Routing Architecture for Real-time CSCW Applications. Technical Report NUCCS-97-16, College of Computer Science, Northeastern University.

Matta, I. and Eltoweissy, M. (1998). A Scalable QoS Routing Architecture For Real-Time CSCW Applications. In *Proceedings Fourth IEEE Real-Time Technology and Applications Symposium*, pages 137–142, Los Alamitos, CA, USA. IEEE Computer Society.

McCanne, S., Brewer, E., Katz, R., Rowe, L., Amir, E., Chawathe, Y., Coopersmith, A., Mayer-Patel, K., Raman, S., Schuett, A., Simpson, D., Swan, A., Tung, T.-L., and Wu, D. (1997). Toward a Common Infrastructure for Multimedia-Networking Middleware. In *Proc. 7th Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 97)*, St. Louis, Missouri. Available at `http://bmrc.berkeley.edu/research/publications/1997/146/146.html`.

McKinley, P., Barrios, R., and Malenfant, A. (1999a). Design and Performance Evaluation of a Java-Based Multicast Browser Tool. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, pages 314–322, Austin, Texas.

McKinley, P., Malenfant, A., and Arango, J. (1999b). Pavilion: A Middleware Framework for Collaborative Web-Based Applications. In *Proceeding of GROUP*, pages 179–188, Phoenix Arizona US. Department of Computer Science and Engineering, Michigan State University, ACM.

Meyer, D. (1998). Administratively Scoped IP Multicast. RFC 2365. IETF.

Miller, C. K. (1997). Reliable Multicast Protocols: A Practical View. In *22nd IEEE Conference on Local Computer Networks (LCN '97)*, pages 369–378, Minneapolis, MN. StarBurst Communications Corporation, IEEE.

Miller, C. K. (1999). *Multicast Networking and Applications*. Addision Wesley, Reading, Massachusetts.

Mills, D. L. (1990). On The Accuracy And Stability Of Clocks Synchronized By The Network Time Protocol In The Internet System. *Computer Communication Review*, **20**(1), 65–75.

Mills, D. L. (1992). Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305, IETF.

Moy, J. (1994a). Multicast Extensions To OSPF. RFC 1584, IETF.

Moy, J. (1994b). Multicast Extensions To OsPF. RFC 1585, IETF.

Musa, J. D., Iannino, A., and Okumoto, K. (1990). *Software Reliability: Measurement, Prediction, Application.* McGraw-Hill publishing Company.

Nandagopal, T., Lee, K.-W., Li, J.-R., and Bharghavan, V. (2004). Scalable Service Differentiation Using Purely End-To-End Mechanisms: Features and Limitations. *Computer Networks*, **44**(6), 813–833.

Nentwig, L., Manhart, S., and Sandkuhl, K. (1996). Hotline And Consulting In A Metropolitan Area Network: The Hotcon Approach To Integrated Services. *Computer Networks and ISDN Systems*, **28**(4), 481–490.

Ngadi, M. and Doherty, B. (2003a). Multi-users Shared Whiteboard under Web-multicasting. In *Proceedings of the International Symposium on Information and Communication Technologies*, volume 1, pages 511–516, Trinity College Dublin, Ireland. Computer Science Department, Aston University, Computer Science Press.

Ngadi, M. and Doherty, B. (2003b). Serverless Web-multicast Chat System for Multi-users. In J. Lovelle, B. Rodriguez, L. Aguilar, and M. Ruiz, editors, *Web Engineering: International Conference, ICWE 2003*, volume 1, pages 519–522, Oviedo, Spain. Computer Science Department, Aston University, Springer.

Ngadi, M. and Doherty, B. (2003c). Web-multicast Chat System for Collaborative Learning. In *Proceedings of the International Conference on e-Society*, volume 1, pages 274–280, Lisbon, Portugal. Aston University, IADIS.

Ngwenyama, O. K. and Lyytinen, K. J. (1997). Groupware Environments As Action Constitutive Resources: A Social Action Framework For Analyzing Groupware Technologies. *Computer Supported Cooperative Work*, **6**(1), 71–93.

Nickless, B. (2003). IPv4 Multicast Unusable Group And Source Addresses. Internet Draft draft-ietf-mboned-ipv4-mcast-unusable-01.txt, IETF.

Orlikowski, W. (1992). Learning from Notes: Organizational Issue in Groupware Implementation. In *Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work*, pages 362–369, Toronto. ACM Press.

Parnes, P., Mattsson, M., Synnes, K., and Schefstroem, D. (1996). mWeb: A Framework For Distributed Presentations Using the WWW and the MBone. In *W3C workshop, Real Time Multimedia and the Web (RTMW '96)*, pages 1–5, Sophia Antipolis, France.

Parnes, P., Mattsson, M., Synnes, K., and Schefstroem, D. (1997). The WebDesk Framework. In *The Seventh Annual Conference of the Internet Society (INET'97)*, Kuala Lumpur, Malaysia. Available at `http://www.cdt.luth.se/~peppar/docs/lic/html/node39.html`.

Parr, G. and Curran, K. (1999). Multiple Multicast Groups For Multimedia On The Internet. *Information and Software Technology*, **41**(2), 91–99.

Patterson, J. F., Hill, R. D., Rohall, S. L., and Meeks, W. S. (1990). Rendezvous: An Architecture for Synchronous Multiuser Applications. In *Proceeding of Computer Supported Cooperative Work (CSCW90)*, pages 317–328, Los Angeles, CA.

Peña Mora, F., Hussein, K., Vadhavkar, S., and Benjamin, K. (2000). CAIRO: A Concurrent Engineering Meeting Environment For Virtual Design Teams. *Artificial Intelligence in Engineering*, **14**(3), 203–219.

Pentagon (1997). *Army Science and Technology Master Plan (ASTMP)*, volume 1 of *Army Document*, chapter Technology Development. Pentagon, Washington.

Poo, G. and Goscinski, A. (1998). Performance Comparison Of Sender-Based And Receiver-Based Reliable Multicast Protocols. *Computer Communications*, **21**(7), 597–605.

Raatikainen, K. (2002). *Research Challenges in Wireless Internet*. Wireless internet presentation, Department of Computer Science, University of Helsinki, Helsingin Yliopisto.

Radoslavov, P., Estrin, D., Govindan, R., Handley, M., Kumar, S., and Thaler, D. (2000). The Multicast Address-Set Claim (MASC) Protocol. RFC 2909. IETF.

Reynolds, J. and Postel, J. (1994). Assigned Numbers. RFC 1700, ISI.

Rodden, T. (1991). A Survey Of CSCW Systems. *Interacting With Computers - The Interdisciplinary Journal of Human-computer Interaction*, **3**(3), 319–353.

Rodriguez, P., Ross, K., and Biersack, E. W. (1998). Improving the WWW: Caching Or Multicast? *Computer Networks and ISDN Systems*, **30**(22-23), 2223–2243.

Roseman, M. and Greenberg, S. (1992). GROUPKIT: A Groupware Toolkit For Building Real-Time Conferencing Applications. In *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work*, pages 43–50. ACM Press.

S. Hanna, B. Patel, M. S. (1999). Multicast Address Dynamic Client Allocation Protocol (MADCAP). RFC 2730, IETF.

Shin, M. K. (1997). Extending the WWW for Multicasting an HTML Document. In *The Seventh Annual Conference of the Internet Society (INET'97)*, Kuala Lumpur, Malaysia. Available at `http://pec.etri.re.kr/~mkshin/publication/inet/index.htm`.

Shin, M. K. and Lee, J. Y. (1998). Web-Based Real-time Multimedia Application for the MBone. In *The Eighth Annual Conference of the Internet Society INET'98*, Geneva, Switzerland.

Shin, M. K., Lee, J. Y., Bae, J. S., and Hahm, J. H. (1998). The RTMW Application : Bringing Multicast Audio/Video to the Web. *Computer Networks and ISDN Systems*, **30**(1-7), 685–687.

Shirazi, J. (2000). *Java Performance Tuning*. O'reilly & Associates, Inc, USA.

Shirmohammadi, S., Oliveira, J., and Georganas, N. (1998). Applet-Based Telecollaboration: A Network-Centric Approach. *IEEE Multimedia Magazine*, **5**(2), 64–73.

Spellman, P., Mosier, J., Deus, L., and Carlson, J. (1997). Collaborative Virtual Work. In *Proceedings of the International ACM SIGGROUp Conference on Supporting Group Work: The Integration Challenge*, pages 197–203. ACM.

Stardust Technologies, I. (1997a). How IP Multicast Work. Technical paper, IP Multicast Initiative (IPMI), Stardust Technologies, Inc., Campbell, USA. Available at `http://www.isel.ipl.pt/~pribeiro/RC2/IPMulticast/howipmcworks.pdf`.

Stardust Technologies, I. (1997b). Implementing IP Multicast in Different Network Infrastructure. Technical paper, IP Multicast Initiative (IPMI), Stardust Technologies, Inc., Campbell, USA. Available at `http://www.isel.ipl.pt/~pribeiro/RC2/IPMulticast/netinfra.pdf`.

Stardust Technologies, I. (1997c). Introduction to IP Multicast Routing. Technical paper, IP Multicast Initiative (IPMI), Stardust Technologies, Inc., Campbell, USA. Available at `http://www.isel.ipl.pt/~pribeiro/RC2/IPMulticast/introrouting.pdf`.

Stardust Technologies, I. (1997d). Writing IP Multicast-enabled Applications. Technical paper, IP Multicast Initiative (IPMI), Stardust Technologies, Inc., Campbell, USA. Available at `http://www.isel.ipl.pt/~pribeiro/RC2/IPMulticast/ipmcapps.pdf`.

Sun Microsystems, I. (1998). *Revitalize the Workgroup Without Replacing the LAN.* Sun Microsystems, Inc., California, USA, suns enterprise workgroup solutions edition.

Takenaka, T. and Sasaki, H. (2004). Adaptive Reliable Multicast Using TCP Protocol In Heterogeneous Environments. *Electronics And Communications In Japan Part I-Communications*, **87**(3), 28–38.

Thaler, D., Handley, M., and Estrin, D. (2000). The Internet Multicast Address Allocation Architecture. RFC 2908, IETF.

Touvet, J.-C. (1996). Multicast Mosaic. Available on-line. `http://www.edelweb.fr/EdelStuff/EdelContrib/jctmcm.html`.

Tseng, W.-Y., Sue, C.-C., and Kuo, S.-Y. (1999). Performance Analysis for Unicast and Multicast Traffic in Broadcast- and-Select WDM Networks. In *The Fourth*

IEEE Symposium on Computers and Communications, pages 72–79, Red Sea, Egypt. Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, IEEE.

Wiil, U., Tata, S., and Hicks, D. (2003). Cooperation Services In The Construct Structural Computing Environment. *Journal of Network and Computer Applications*, **26**(1), 115–137.

Wilde, E. (1994). Supporting CSCW Applications With An Efficient Shared Information Space. Technical Report TIKrep, Computer Engineering and Networks Laboratory (TIK), ETH Zrich, Switzerland.

Wilson, P. (1991). *Computer Supported Cooperative Work: An Introduction*, volume 1. Intellect Book.

Wittmann, R. and Zitterbart, M. (2001). *Multicast Communication - Protocols, Programming, and Applications*. Academic Press, San Diego, California.

Wolski, R. (1998). Dynamically Forecasting Network Performance Using the Network Weather Service. *Cluster Computing*, **1**(1), 119–132.

Wright, L. K., McCanne, S., and Lepreau, J. (2000). A Reliable Multicast Webcast Protocol For Multimedia Collaboration And Caching. In *Proceedings of The Eighth ACM International Conference on Multimedia*, pages 21–30. ACM Press.

Yavatkar, R., Griffoen, J., and Sudan, M. (1995). A Reliable Dissemination Protocol For Interactive Collaborative Applications. In *Proceedings of the Third ACM International Conference on Multimedia*, pages 333–344. ACM Press.

Zabele, S., Rohall, S. L., and Vinciguerra, R. L. (1994). High Performance Infrastructure For Visually-Intensive CSCW Applications. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 395–403. ACM Press.

Appendix A

Research Planning

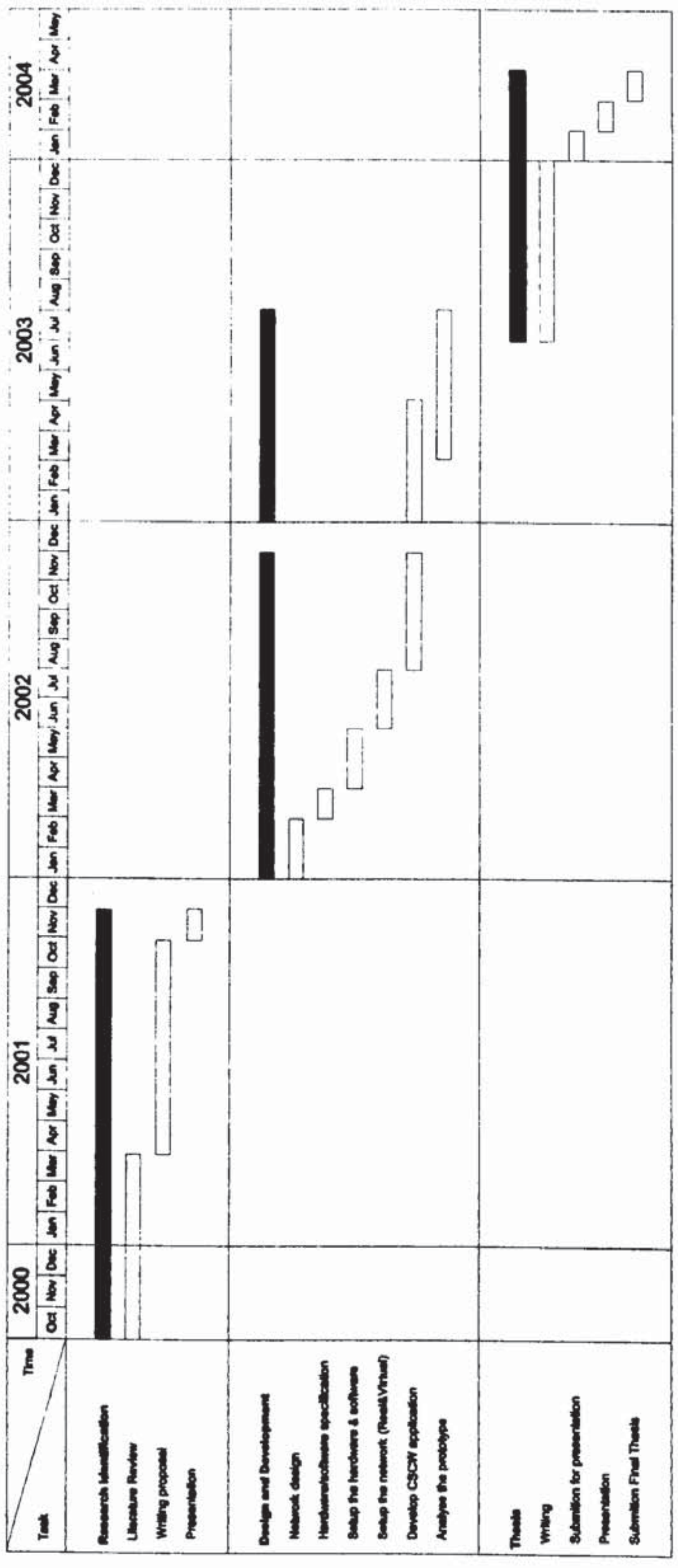


Figure A.1: Research Planning

Appendix B

Distributed Architecture

There are a number of different distributed architecture available to implement application on network. The architecture are (Zabele *et al.*, 1994; Balter *et al.*, 1995):

- **centralised architecture**

  In this architecture, all participants access one central server to operate. All information is located in the central server. The central server is responsible for all processes such as updating changes of information, processes user requests, and information flow recovery. For example, Rendezvous (Patterson *et al.*, 1990) is a CSCW application implemented using centralised architecture. This architecture also known as client/server architecture.

- **replicated architecture**

  In this architecture, each participant accesses or executes an instance of a shared application that is located on a server nearest to them. Inputs from each participant is distributed from the user window to all instances and the output from each copy is delivered only to the local window system. The advantages are high robustness, short response time, and better support for heterogemeity and view customization. However, the main drawback of this scheme is hard consistency management. Several toolkits have adopted this architecture such as MMconf (Crowley *et al.*, 1990) and GroupKit (Roseman and Greenberg, 1992).

- **peer-to-peer architecture**

Peer-to-peer sometimes known as P2P. A peer-to-peer architecture allows hardware or software to function on a network without the need for central servers. Peer-to-peer is commonly used in configuring home computer networks where the cost of a server can be difficult to justify. The approach has also been popularized by some Net software applications such as Groove and Napster. The P2P acronym has acquired a non-technical meaning as well. Some people have described this second meaning as "people-to-people." From this angle, P2P is a model for developing software and growing businesses that help individuals on the Net to meet each other and share common interests.

- **hybrid architecture**

  Hybrid architecture is a network architecture that consists of two or more architectures. For example, if a client/server and a distributed model merge to make another model, the new model is referred to a hybrid model. The hybrid architecture will acquire capabilities and advantages of each architecture that constructed as a hybrid architecture. For example, a hybrid architecture of replicated and centralised architectures. The idea is to replicate user interface connected to a central application process. Then, user interface operations are performed locally and all other tasks are performed by centralised application process.

In the first approach, management is easy but bottlenecks happen when there are many participants and large sizes of data involved in the communication processes. In the second approach, bottlenecks may reduce but replication processes increase while processing real-time data such as video and audio conferencing. In the third approach, the bottlenecks may reduce but management process complicated. The hybrid architecture is used to overcome the disadvantages of architectures mentioned above. Thus, hybrid architecture depends on selected component of architectures, provides easy management, less bottlenecks, and reduced replication processes. Most of the CSCW applications are implemented by the unicasting method either in a client/server model or a distributed model. These two communication methods, unicasting and multicasting, are different on their communication, addressing, and performance, which effected the architecture, design, and development of applications. This study however, has used a different approach by using the multicasting method in a hybrid architecture.

Appendix C

Multicasting

C.1 Introduction

This appendix explains briefly concepts of multicasting, requirements, and multicast protocols. The detail of information can be found in different documents as follow:

- information on how multicast works from Stardust Technologies (1997a)
- information on multicast routing from Stardust Technologies (1997c)
- information on how multicasting work on different network from Stardust Technologies (1997b), and
- information on how to enable multicast for application from Stardust Technologies (1997d)

C.2 Overview

Multicasting (Deering, 1989) is one method that allows distribution of data from one sender to many receivers. A multicast datagram is delivered to all members of its destination host group with the same "best-efforts" reliability as regular unicast IP datagrams. The membership of a host group is dynamic i.e. hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. Allowing any host to "join" or "leave" at any time, makes multicasting easy to scale to a large number of participating hosts. The extendibility of the group size makes the IP multicasting model very attractive from the perspective of scalability (Hardjono and Cain, 1999).

In addition, at the application level, a single group address may have multiple data streams on different port numbers, on different sockets, in one or more applications. Multiple applications may share a single group address on a host. The distribution in multicasting is made through a multicasting distribution tree. The multicasting distribution tree is shaped using a multicasting routing protocol such as DVRMP, CBT, MOSPF and PIM-SM. In multicasting, any host can "join" or "leave" a multicasting group using a group membership protocol such as Internet Group Membership Protocol (IGMP) (Deering, 1989; Cain *et al.*, 2002).

C.3 IP Multicast Requirements

To support multicasting, the sending and receiving nodes and network infrastructure between them must be multicast-enabled, including intermediate routers. Requirements for multicasting at the end node hosts are (Stardust Technologies, 1997b):

- Support for IP Multicast transmission and reception in the TCP/IP protocol stack.
- Software supporting IGMP to communicate requests to join a multicast group(s) and receive multicast traffic.
- Network interface cards which efficiently filter LAN data link layer addresses mapped from network layer IP Multicast addresses.
- IP Multicast application software such as video conferencing.

To run or evaluate IP Multicast on a LAN, only the above are needed; no routers need be involved for a host's adapter to create or join a multicast group and share multicast data with other hosts on that LAN segment. To expand IP Multicast traffic to a WAN requires:

- All intermediate routers between the sender(s) and receiver(s) must be IP Multicast-capable. Many new routers have support for IP Multicast; older ones may require memory before they can be upgraded.
- Firewalls may need to be reconfigured to permit IP Multicast traffic.

IP Multicast has broad and growing industry backing, and is supported by many vendors of network infrastructure elements such as routers, switches, TCP/IP stacks, network interface cards, desktop operating systems and application software.

Figure C.1: Multicast-enable Components (Stardust Technologies, 1997b)

Figure C.1 shows at a high level, components that must be multicast-enabled. The direction of traffic shown is for multicast datagrams. Traffic needed to communicate host group membership and routing information is not shown. IP tunneling is an interim mechanism used to connect islands of multicast routers separated by links which do not support IP Multicast. With this approach, multicast datagrams are encapsulated in a standard point-to-point unicast datagram. Tunneling is used extensively in the MBone.

C.4 IP Multicast Addressing

Multicast IP uses Class D IP addresses for multicast transmission to a group. Each group simply have one address space indicating a multicast group. The addresses used only on a session-by-session basis. The multicast address space occupies the range from 224.0.0.0 to 239.255.255.255. A common problem of all multicast applications is how to choose the multicast group address. This address has to be selected very carefully. If more than one application uses the same group, both data streams may interfere, or they may unnecessarily stress the network connections as well as the end systems. Three different types of address allocation are defined (Wittmann and Zitterbart, 2001):

- static allocation
- scope-related allocation
- dynamic allocation

For some well-known applications, the Internet Assigned Numbers Authority (IANA) has reserved single static addresses (Albanna *et al.*, 2001) and a few multicast applications have chosen other statically defined multicast groups on their own which are no longer available to other applications (Nickless, 2003). The Internet Assigned Number Authority (IANA) maintains lists of registered users and assigns new numbers for new uses. Examples of addresses, as assigned by the Internet Assigned Numbers Authority (IANA), are 224.0.0.1, the "all-hosts group" used to address all IP Multicast hosts on the directly connected network, and 224.0.0.2, which addresses all routers on a LAN. The range of addresses between 224.0.0.0 and 224.0.0.255 is reserved for routing protocols and other low-level topology discovery or maintenance protocols. The remaining multicast addresses, 224.0.1.0 to 239.255.255.255 are either assigned to various multicast applications or currently unassigned. Other addresses and ranges have been reserved for applications, such as 224.0.13.0 to 224.0.13.255 for Net News. The set from 239.0.0.0 to 239.255.255.255 is reserved for various "administratively scoped" applications, which do not necessarily have an Internet-wide scope. These reserved IP Multicast addresses are listed in RFC 1700 (Reynolds and Postel, 1994).

Table C.1: Administrative Regions in Multicast Address Space (Wittmann and Zitterbart, 2001)

Additionally, the multicast address range (class D, 224.0.0.0/24) has been divided into a number of administrative regions shown in Table C.1. Packets with a link-local address will not be forwarded by any router. Administratively scoped addresses are used in a single organisation (Meyer, 1998) and only packets with an address in the global scope range are forwarded through the entire Internet. In order to have unique multicast addresses available for individual organizations and routing domains, several approaches have been initiated. One example is the multicast-scope zone announcement protocol (MZAP) (Handley *et al.*, 2000a). This protocol allows a dynamic allocation of multicast address space. The term session is used to describe an application involving one or more multicast groups for its data communication. Most multimedia applications, such as video conferencing, use the session announcement protocol (SAP)(Handley *et al.*, 2000b) in conjunction with the session directory (SDR)(Handley, 1995) to choose and to announce multicast groups for their transmissions. The SDR tool chooses some unused multicast groups to initiate a session. This session, for example a video conference can be joined by any user. The shared usage of a single multicast address by more than one application can be prevented by using this mechanism. Another benefit of using the SDR is its capability to provide a set of data about each session. Examples are the name and the purpose of the session, the time when the session becomes active and the type of media streams used in this session. The SDR announces all this information in addition to the chosen multicast addresses by using the session description protocol (SDP)(Handley and Jacobson, 1998; Handley *et al.*, 2002). The current approach is to provide a set of protocols to choose globally unique multicast addresses. The multicast address allocation architecture (Thaler *et al.*, 2000) consists of local Multicast Address Allocation Servers (MAASs) and three protocols (Wittmann and Zitterbart, 2001):

- a host-server protocol, which allows a client to request an address from its local MAAS: Multicast Address Dynamic Client Allocation Protocol (MADCAP)(S. Hanna, 1999),
- an intra-domain-server protocol, which is used to claim addresses and to inform local peer MAASs about used multicast addresses: Multicast Address Allocation Protocol (MAAP)(Handley, 1997), and
- an inter-domain protocol, used to allocate multicast address sets to domains: Multicast Address-Set Claim (MASC)(Kumar *et al.*, 1998; Radoslavov *et al.*, 2000).

C.5 Group Management

The group management is responsible for two main tasks:

- communication between the end system and local router, and
- selection of a feasible group address

The first one, the communication between the end system and the local router, is done by using IGMP (Internet Group Management Protocol). Unfortunately, there is to date no specific criterion for the selection of the group address. However, some known approaches are analysed.

C.5.1 Internet Group Management Protocol (IGMP)

The Internet Group Management Protocol (Deering, 1989; Cain *et al.*, 2002) is responsible for the communication between end systems and locally connected routers. The routers require information about the group membership of their directly connected hosts to compute their multicast forwarding tables and to provide a network wide multicast connectivity. There are three types of communication in IGMP:

- maintain a group: the router performs a so-called query-response process every 60 sec.
- to update its information about current multicast memberships.
- join a group: a host informs its routers that it wants to receive traffic for a particular multicast group

In addition, at the application level, a single group address may have multiple data streams on different port numbers, on different sockets, in one or more applications. Multiple applications may share a single group address on a host. The distribution in multicasting is made through a multicasting distribution tree. The multicasting distribution tree is shaped using a multicasting routing protocol such as DVRMP, CBT, MOSPF and PIM-SM. In multicasting, any host can "join" or "leave" a multicasting group using a group membership protocol such as Internet Group Membership Protocol (IGMP) .

Figure C.2: IGMP (Stardust Technologies, 1997a)

Multicast packets from remote sources must be relayed by routers, which should only forward them on to the local network if there is a recipient for the multicast host group on the LAN. The Internet Group Management Protocol (IGMP) is used by multicast routers to learn the existence of host group members on their directly attached subnets (refer Figure C.2). It does so by sending IGMP queries and having IP hosts report their host group memberships. The basic version of IGMP is described in RFC 1112.

IGMP is loosely analogous to ICMP and is implemented over IP. IGMP messages are encapsulated in IP datagrams. IGMP has only two kinds of packets: Host Membership Query and Host Membership Report, with the same simple fixed format containing some control information in the first word of the payload field and a class D address (Figure C.3) in the second word:

Figure C.3: Class D Multicast Address (Stardust Technologies, 1997a)

To determine if any hosts on a local subnet belong to a multicast group, one multicast router per subnet periodically sends a hardware (data link layer) multicast IGMP Host Membership Query to all IP end nodes on its LAN, asking them to report back on the host groups memberships of their processes. This query is sent to the all-hosts group (network address 224.0.0.1) and a TTL of 1 is used so that these queries are not propagated outside of the LAN. Each host sends back one IGMP Host Membership Report message per host group, sent to the group address, so all group members see it (thus only one member reports membership).

When a process asks its host to join a new multicast host group, the driver creates a hardware multicast address, and an IGMP Host Membership Report with the group address is immediately sent. The host's network interface is expected to map the IP host group addresses to local network addresses as required to update its multicast reception filter. Each host keeps track of its host group memberships, and when the last process on a host leaves a group, that group is no longer reported by the host.

Periodically the local multicast router sends an IGMP Host Membership Query to the "all-hosts" group, to verify current memberships. If all member hosts reported memberships at the same time frequent traffic congestion might result. This is avoided by having each host delay their report by a random interval if it has not seen a report for the same group from another host. As a result, only one membership report is sent in response for each active group address, although many hosts may have memberships.

IGMP updates are used by multicast routing protocols to communicate host group memberships to neighboring routers, propagating group information through the internetwork. IGMP is used to identify a designated router in the LAN for this purpose. The bandwidth needed to transmit host group information is usually slight compared to the multicast application traffic, so this propagation method is workable. More sophisticated methods enable routers to determine dynamically how to best forward the multicast application traffic, as discussed in the next section.

C.6 Sending IP Multicast Datagrams

To send an IP Multicast datagram, the sender specifies an appropriate destination address, which represents a host group. IP Multicast datagrams are sent using the same "Send IP" operation used for unicast datagrams. Compared to sending of IP Multicast datagrams, reception of IP Multicast datagrams is much more complex, particularly over a WAN.

To receive datagrams, a user's host application requests membership in the multicast host group associated with a particular multicast. This membership request is communicated to the LAN router and, if necessary, on to intermediate routers between the sender and the receiver. As another consequence of its group membership request, the receiving host's network interface card starts filtering for the LAN-specific hardware (data-link layer) address associated with the new multicast group address. WAN routers deliver the requested incoming multicast datagrams to the LAN router, which maps the host group address to its associated hardware address and builds the message (for example, an Ethernet frame) using this address. The receiving host's network interface card and network driver, listening for these addresses, pass the multicast messages to the TCP/IP protocol stack, which makes them available as input to the user's application, such as a video viewer.

Whereas an IP unicast address is statically bound to a single local network interface on a single IP network, an IP host group address is dynamically bound to a set of local network interfaces on a set of IP networks. An IP host group address is not bound to a set of IP unicast addresses. Multicast routers don't need to know the list of member hosts for each group. A multicast router attached to an Ethernet need associate only a single Ethernet multicast address with each host group having a local member.

Appendix D

Multicast Routing Protocols

This appendix explains multicasting routing protocols. There are two different routing protocols for multicasting: intra-domain routing and inter-domain routing. Intra-domain routing means routing in a limited sized network. The expectations for such routing protocols are very different from those for inter-domain routing protocols. Typically, the best path between two nodes is defined by numerous criteria including the number of inter-node hops, the most available bandwidth, and the smallest. In inter-domain routing, a best path means the path with the lowest cost.

D.1 IP Routing

The Internet is composed of a myriad of subnetworks connected by routers. When the source of a message is located on one subnet and the destination is located on a different subnet, there must be some way of determining how to get from the source to the destination. This is the function of the IP Protocol. Each host on the Internet has an address that identifies its physical location; part of the address identifies the subnet on which it resides and part identifies the particular host on that subnet. Routers periodically send routing update messages to adjacent routers, conveying the state of the network as perceived by that particular router. This data is recorded in routing tables that are then used to determine optimal transmission paths for forwarding messages across the network.

Unicast transmission involves transmission from a single source to a single destination. Thus, the transmission is directed towards a single physical location that is specified by the host address. The routing procedure, as described above, is relatively straightforward because of the binding of a single address to a single host.

D.1.1 Multicasting Routing

Routing multicast traffic is a more complex problem. A multicast address identifies a particular transmission session, rather than a specific physical destination. An individual host is able to join an ongoing multicast session, by using IGMP to communicate this desire to its subnet router. A naive approach to sending data to multiple receivers would be for the source to maintain a table identifying all the receiving subnets participating in the session and to send a separate copy of the data to each receiving subnet. However, this would be an extremely inefficient use of bandwidth, since many of the data streams would follow the same path throughout much of the network.

New techniques have been developed to address the problem of efficiently routing multicast traffic. Since the number of receivers for a multicast session can potentially be quite large, the source should not need to know all the relevant addresses. Instead the network routers must somehow be able to translate multicast addresses into host addresses. The basic principal involved in multicast routing is that routers interact with each other to exchange information about neighboring routers. To avoid duplication of effort, a single router is selected (via IGMP) as the Designated Router for each physical network.

D.1.2 Spanning Trees

For efficient transmission, Designated Routers construct a spanning tree that connects all members of an IP Multicast group. A spanning tree has just enough connectivity so that there is only one path between every pair of routers, and it is loop-free. If each router knows which of its lines belong to the spanning tree, it can copy an incoming

Figure D.1: Spanning Tree (Stardust Technologies, 1997a)

multicast datagram onto all of its outgoing branches, generating only the minimum needed number of copies. Messages are replicated only when the tree branches, thus minimising the number of copies of the messages that are transmitted through the network.

Since multicast groups are dynamic, with members joining or leaving a group at any time, the spanning tree must be dynamically updated. Branches in which no listeners exist must be discarded (pruned). A router selects a spanning tree based on the network layer source address of a multicast packet, and prunes that spanning tree based on the network layer destination address. The spanning algorithm used and how multicast routers interact depends on the objectives of the routing protocol.

D.1.3 Two Basic Approaches to IP Multicast Routing

IP Multicast routing algorithms and protocols generally follow one of two basic approaches, depending on the distribution of multicast group members throughout the network. The first approach is based on the assumption that the multicast group members are densely distributed throughout the network and bandwidth is plentiful, i.e., almost all hosts on the network belong to the group. So-called "dense-mode" multicast routing protocols rely on periodic flooding of the network with multicast traffic to set up and maintain the spanning tree. Dense-mode routing protocols include Distance Vector Multicast Routing Protocol (DVMRP), Multicast Open Shortest Path First (MOSPF), and Protocol-Independent Multicast - Dense Mode (PIM-DM).

The second approach to multicast routing is based on the assumption that the multicast group members are sparsely distributed throughout the network and bandwidth is not necessarily widely available, for example across many regions of the Internet. It is important to note that sparse-mode does not imply that the group has a few members, just that they are widely dispersed. In this case, flooding would unnecessarily waste network bandwidth and hence could cause serious performance problems. Hence, "sparse-mode" multicast routing protocols must rely on more selective techniques to set up and maintain multicast trees. Sparse-mode routing protocols include Core Based Trees (CBT) and Protocol-Independent Multicast - Sparse Mode (PIM-SM).

D.1.4 Other Protocols which use IP Multicast

There are a number of protocols presently being developed by the Internet community, IETF working groups and industry vendors to support new applications of IP Multicast. RTP, the Real-time Transport Protocol, provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RSVP, the ReSerVation Protocol, enhances the current Internet architecture with support requests for a specific quality of service (QoS) from the network for particular data streams or flows. RTSP, the Real-Time Streaming Protocol is an application-level protocol for control over the delivery of data with real-time properties to enable controlled, on-demand delivery of real-time data, such as audio and video. Reliable multicast protocols are being developed to overcome the limitations of unreliable multicast datagram delivery and expand the uses of IP Multicast.

D.2 Multicasting Routing Protocol in the Internet

Deering's work (Deering, 1990) in the late 1980s is considered as a pioneering in Internet multicasting. He defined a service model for multicasting and devised the related algorithms. Multicasting routing protocols were used at a router to optimally route multicasting packets through the network or internetwork from the source to multiple destinations that consist of the members of the multicasting group. This approach is similar in concept to the unicast (point-to-point) routing protocols (Miller, 1999). According to Lin and Chang (1999), there are five mainstream multicasting protocol in the Internet:

1. Distance Vector Multicasting Routing Protocol(DVMRP) (Deering *et al.*, 1988)

2. Multicasting extensions to OSPF (MOSPF) (Moy, 1994a,b)

3. Core Based Tree (CBT) (Ballardie, 1995)

4. Conference Steiner Multicasting(CSM) (Aggarwal *et al.*, 2000)

5. Protocol-Independent Multicasting-Sparse Mode (PIM-SM) and Dense Mode(PIM-DM) (Deering *et al.*, 1997; Estrin *et al.*, 1997)

Different protocols use different techniques to construct the multicasting delivery tree and each of them has its special functions and properties. Multicasting routing protocols build delivery trees which are either source-specific or center-specific. As shown in Figure D.2, Distance Vector Multicasting Routing Protocol (DVMRP), Multicasting Extensions to Open Shortest Path First (MOSPF), and Protocol Independent Multicasting-Dense Mode (PIM-DM) build source-specific trees; Center Based Trees (CBT), Protocol Independent Multicasting-Sparse Mode (PIM-SM) and Conference Steiner Multicasting(CSM) construct center-specific trees.

Unicast routing protocols use one of two basic techniques: *distance vector* or *link state*. While in multicasting routing protocol use either *distance vector* or *link state* or *shared trees*. In *dense-mode* multicasting routing protocols, they require some form

Figure D.2: Multicasting Routing Protocol Family Tree (Miller, 1999)

of flooding of datagrams to the network to find multicasting routers. This approach is suitable for areas with dense concentration of group members, such as campus networks. *Sparse-mode* multicasting routing protocols are the best suited for widely dispersed group members in a wide are network. All multicasting routing protocols must set up distribution tree to route datagrams to the members of the group in an optimal way. The challenge is to create multicasting routing protocols that can set up these distribution trees quickly, efficiently, and without excessive network traffic.

Appendix E

Reliable Multicast Protocols

E.1 Introduction

This appendix describes briefly on reliable multicast transport protocol. Two protocols are describes in this appendix which related to the study.

E.2 Classification of protocols

With the growing interest of network computing such as the global Internet, the requirement of reliable multicast protocols in group communication is ever increasing. Group communication is needed in inactive group collaborative works. These include applications that support video, voice, data, and shared whiteboards over wide area networks (Poo and Goscinski, 1998). There are two main classes of reliable multicast transport protocols based on the type of loss recovery mechanism. The first class uses negative-acknowledgements from the receiver to trigger retransmissions from the source and the other relies on positive ACKs. These approaches to providing reliable and scalable multicast communication also known as sender-based and receiver-based.

The sender-based approach is the traditional method. In this approach, the sender multicasts data to receivers and maintains the state information regarding all receivers using sequence numbers. Reliability is ensured by requiring the receivers to return positive acknowledgments (ACKS) for all packets correctly received and the use of timers at the sender to retransmit lost/damaged packets.

The second is the receiver-based approach. This approach shifts the responsibility for reliable data delivery to the receivers. Each receiver is responsible for detecting lost/damaged packets itself using sequence numbers. It raises repair request(RR) packets when it requires the sender to retransmit a packet. The repair request packet can be issued over point-to-point channels. Alternatively, it can be multicast with the introduction of a random delay before transmission. The latter further improves the efficiency of the system.

E.3 Scalable Reliable Multicast

Scalable Reliable Multicast (SRM) (Floyd *et al.*, 1997) is a reliable multicast framework for application level framing and light-weight sessions. SRM is a NACK-based (negative acknowledgement), fully-decentralised reliable multicast protocol originally described by Floyd *et al.* (1997). The algorithms of this framework are efficient, robust and scale well to both very large networks and very large sessions. The SRM framework builds on Clark and Tennenhouses principle of Application Level Framing (ALF) (Floyd *et al.*, 1997), which provides solution to the problem of reliable-multicast API design because its flexibility offers applications the opportunity to actively participate in the loss-recovery procedure.

To avoid ACK-implosion, SRM uses NACKs. Receivers detect losses from discontinuities in sequence numbers (or by other means with a generic data naming scheme) and transmit NACKs as a request for retransmission of the lost data 1. A randomised algorithm determines when a receiver transmits a NACK. These NACKs are multicast to the entire group so that any receiver, in particular the closest receiver with the requested data, may generate a repair in response to a NACK. The repair messages are also multicast to the entire group, so that all receivers that missed that packet can be repaired by a single response. The repair message traffic likewise makes use of the randomized timer algorithm. To avoid NACK implosion, receivers that observe a NACK for data that they too have not received do not send their own NACK 2 and await the repair data. The goal of the randomized NACK transmission algorithm is to minimize the number of duplicate NACK messages sent. To accomplish this, each receiver delays the transmission of a NACK by an amount of time given by the expression

backoff = D.(C1 + C2 r)

where

backoff is the amount of delay,

D is an estimate of the one-way delay from the receiver to the source that generated the lost data packet,

$C1$; $C2$ are non-negative protocol constants, and

r is a uniformly distributed random number in [0; 1]

This random delay provides receivers with the opportunity to suppress the transmission of similar pending NACKs; that is, delaying the transmission of NACKs by a random amount increases the likelihood that a NACK from one receiver is delivered to another receiver before that receiver sends its own NACK, and thus, reduces the total number of NACKs. The deterministic-delay component induces suppression effects across receivers situated at varying distances from the point of loss (e.g., a chain topology), while the random-delay component induces suppression effects across receivers situated at equal distances from the point of loss (e.g., a star topology).

The framework has been prototyped in wb, a distributed whiteboard application, and has been extensively tested on a global scale with sessions ranging from a few to more than 1000 participants. The framework can be summarized as follows: There are basically three types of packets: heartbeats, NACKs and repairs. Each member periodically sends out a heartbeat including the sequence number of the latest packet it has sent. These heartbeats are used by the other members to detect packet loss by comparing the sequence number in the heartbeat and the sequence number of the last data-packet received. If a packet is lost a negative acknowledgment (NACK) is sent to all members using the same method of transportation as the original data.

As all members see these NACKs, everyone can participate in the repair of the traffic. This is important because it eases the burden on the original sender and makes the repair process faster if the distance between the NACK sender and the original sender is large. Each member of the session cache the latest data-packets received (and sent) and if they see a NACK for a packet they have in their cache, they retransmit that packet to the whole group as a repair. To minimise the number of NACKs and repairs sent these two operations are preceded by exponential back-off meaning that instead of sending the packet directly, the client waits for a random time (based on the distance to the related other party) and if another client sends a corresponding packet it withdraws its own potential transmission.

E.4 Tree-based Reliable Multicast Protocols

Some schemes including Tree-based Reliable Multicast (TRAM) (Kandansky *et al.*, 2000) and Tree-based Multicast Transport Protocol (TMTP) (Yavatkar *et al.*, 1995) take a different approach to loss recovery. Here, the multicast session members are organised into a transport/session-level hierarchy and attempt to reduce the scale of feedback traffic by localising it to smaller scopes. TMTP combines both sender and receiver initiated techniques. Participants are organized into domains with a single domain manager responsible for error recovery and local retransmission in each domain. Error control at the sender utilizes periodic unicast ACKs from domain managers, time-outs and retransmissions. Domain managers send ACKs upon receipt of a multicast packet. To signal a missing packet, domain members multicast a NACK to the domain manager in combination with NACK suppression. Localised error control is supported by limited scope multicasting via the IP time-to-live (TTL) field. RMTP and TRAM are intended for bulk data transport. RMTP was the earliest tree-based protocol and used statically established hierarchy. Sending periodic ACKs also allows for performing congestion control at the sender. The semantics of congestion control in this case are slowest rate semantics, where the entire session operates at the bandwidth of the slowest network path. TRAM uses dynamic trees to implement local error recovery using ACKs and to scale to a large number of receivers without seriously impacting the sender. Here too, as in LBRM, the administrative overheads in establishing and maintaining the transport-level hierarchy. TMTP uses only end system nodes to construct this hierarchy and attempts to automate the process using monitored loss rates between receivers. However, the stability of such a dynamic system is has not been studied under realistic network conditions.

Appendix F

Light-weight Reliable Multicast Protocol

F.1 Introduction

This appendix explains LRMP in detail which following Liao (2003).

F.2 Motivation

The design of LRMP resulted from the need for a reliable multicast transport protocol in the implementation of a Web multicast tool. Some existing Web multicast tools use a repeated transmission scheme, i.e., the holder of Web documents periodically sends HTML files and hopes that receivers can get a correct copy from one of the repeated transmissions. This scheme uses excessive network bandwidth for potentially unnecessary retransmissions. Theoretically this scheme will converge only after a number of cycles.

A reliable multicast transport protocol is need to deal with packet loss caused by using unreliable transport services, such as UDP. After searching the literature a number of related works have been found. However no implementation in Java was found. A Java program was written to implement the LRMP protocol. The implementation is ideally general-purpose for better reuse, efficient, to meet the needs of a variety of applications, and light-weight so it can be easily embedded in any applications.

As initially this protocol is for use in a Web multicast application, high performance, such as high throughput and low latency is not a main aim. Some delay is acceptable to get reliable and ordered data delivery service. The implementation of the protocol should not introduce heavy control traffic and high protocol overheads.

Attention has quickly focused on SRM for its simplicity and efficiency. Each participant in a group has a symmetrical role, just as in the Ethernet CSMA/CD protocol. One participant does not depend on another participant if it is a simple receiver. Consequently, the failure at one participant does not much influence quality of service for other participants. This is particularly suitable for loosely coupled groups where participants can frequently join and leave.

While SRM is attractive in general, it has some drawbacks. In SRM, when packet loss occurs, all participants are involved in the repair process in hope that the closest site to the loss reporter could send the repaired packet. This scheme reduces scalability, because in case of large group size it will generate considerable control traffic simply for measuring the round trip time between one participant to all other participants. Other aspects it does not address include totally ordered packet delivery and flow control.

A light-weight reliable multicast protocol based on SRM is proposed to meet the design requirement and reduce difficulties. This protocol could be viewed as a modified and improved SRM. It provides basically reliable, totally ordered data packet delivery service and rate-based flow control. LRMP is light-weight, simple to implement and easy to use protocol.

F.3 Design Goals

LRMP should meet the requirements of group communication applications on the MBONE which have loose real time constraints and need reliable and totally ordered data delivery service. By loose real time constraints, we mean that transmission delay is not critical for applications. In another extreme case, real time audio and video conference applications need low transmission delay in order for that audio and video data could be played in a synchronised fashion. By reliable data delivery service, we mean what is sent by a user will be exactly what is received at other users in a group, in the same order and with the same bits. Note that in case of multiple senders, there will be multiple ordered data streams which are generally orthogonal. The main design goals of LRMP can be summarised as:

- Scalable.

  LRMP should be scalable not only to large group size but also to WAN based applications. Since receivers may be geometrically dispersed over the world, the quality of their network connections could be very mixed, which must be taken into account to ensure the normal behavior of a multicast session.

- Light-weight.

  Due to real time and efficiency constraints, the protocol should be low-overhead and have a low control traffic. In particular, for simple receivers (which do not send data) it does not require much CPU power and data buffer area in order to be able to run on low-end machines.

- General purpose.

  It means that no application semantics will be included in the transport protocol. Besides, LRMP is intended to be reusable and embeddable in other applications other than multicast of Web documents.

- Reasonable performance.

  While LRMP is not aimed at providing high performance, the data throughput and transmission delay should be reasonable compared with the quality of service of the underlying network. LRMP as designed initially is not suitable for use with real time continuous media applications.

F.4 Protocol Details

This section presents the algorithms used in LRMP for loss report and repair, failure processing and flow control. LRMP only supposes that underlying network layers are able to deliver data to a group of destinations with indication of the length of packet and source address. In LRMP, receivers do not send ACK packets to acknowledge the receipt of data packets, instead receivers only send NACK to ask the retransmission of missing packets.

F.4.1 Data Encapsulation

In LRMP, application data are just sent as standard RTP data packets. It is the application's responsibility to use a specific payload format to encapsulate media data and fill the timestamp field of packet header. In the RTP header, the extension bit field is not used and is always set 0. In addition, LRMP does not allow the mixer function of RTP, so the contributing source count is always set to 0 and the RTP header has a constant length, 12 bytes. In total, there is a 40 byte overhead for data packets including RTP header (12 bytes), UDP header (8 bytes) as well as the IP header (20 bytes).

F.4.2 Loss Report

A receiver detects packet loss either by checking if there is a gap between the sequence numbers of two successive received packets or the sequence number given by a SYNC packet does not correspond to the last received packet. When packet loss is detected, the receiver does not send immediately a NACK packet, but schedules a timer with a random value to prevent possible NACK-implosion. The initial value of timer is in the following range,

$R0 = (2 * Td, 4 * Td)$

where Td is the distance between the receiver and the sender which is estimated as roughly half the round-trip time.

When the timer expires, if the lost packet(s) are received, the repair process terminates. If a NACK packet for the same loss is received but the lost packet(s) not, the receiver does not send NACK but schedules a new timer. If neither the lost packet(s) nor NACK packet are received, the receiver sends a NACK packet and schedules a new timer to keep the repair process alive. Each time a new timer is scheduled for the same loss event, the receiver clears NACK history and the timer value performs an exponential backoff, i.e., the random value range will become,

$Rn = 2 * Rn - 1$, where n = 1, 2, 3, ...

When the new timer expires, the receiver repeats the above steps. The maximum timer value is set to 30 seconds. When the number of timeouts becomes greater than 16, the repair process is abandoned and a reception failure event is generated.

F.4.3 Sending Repair

When a packet is lost at a receiver, it is very likely that other receivers lose it too. Thus repair packets are sent to the group address with the same TTL (Time To Live) as normal data packets. In SRM, when packet loss occurs, all participants are involved in the repair process in the hope that the closest site to the loss reporter could send the repair.

First for this scheme to work, each participant needs to know the round trip time to all other participants in a group. Consider the case that there are already one thousand participants and a newcomer just joins the group. With optimisation, this new participant requires at minimum one query packet and one thousand responses to determine the round trip time without counting retransmission. If still new comers arrive, the generated control traffic will be considerable.

Second the round-trip time between two sites is generally time-variant since network congestion varies from time to time, in particular over wide area networks. So measurements need to be performed periodically. As control traffic is generally limited to a low percentage of data traffic, (e.g. 5% in RTP), it is hard to do it.

Finally it is likely that duplicated repair packets will be sent to the whole group, just like duplicated NACKs. The former could be avoided and the latter cannot all be eliminated.

The view point is that, an all involved repair scheme is appropriate only if local recovery can be used. In the local recovery scheme, close users (e.g. in a region) are grouped together as subgroups, repair packets are sent in the subgroup when possible by adjusting TTL (time to live) in multicast packets. This scheme will be included in the future version of LRMP, but is not used in this research.

To simplify the problem, in LRMP, only a sender is allowed to send repair packets and only needs to know the round-trip time from the sender to receivers. An additional advantage of this scheme is that receivers do not need to cache received packets if they are in order, and thus require less system resources.

To be application independent, an LRMP sender keeps recently sent packets in a send window. The size of the send window, depending on the used transmission rate, should be enough large to allow loss repair. Liao (1998) recommend that the send window size should correspond to 30 seconds - 2 minutes of data packets at the full specified transmission rate (e.g., between 240 KB and 960 KB at 64 kb/s). As it is possible that repair packets are lost, when a packet is dropped from the send window, if it is resent as repair, it will be put to a resend window which is of a much smaller size. This is to allow any further resends.

Upon receipt of a NACK packet, the sender checks if it is a duplicated NACK. Two resend requests of the same packet is treated as duplicated if they are received within 500 milliseconds. If it is not a duplicated NACK, the sender sends immediately the repair packets. The sender gives higher priority to sending repair packets than normal packets.

F.4.4 Processing Reception Failures

Reception failure occurs when a participant cannot synchronise with the data stream sent by another participant. Reception failures are mainly caused by network partition or a receiver with a very bad link. They occur in the following conditions:

- when the number of timeouts for a loss event (NACK timer) reaches a threshold, repair packets are still not received.
    - there is a too large a difference between the currently received sequence number and the expected sequence number.
    - there is a too large a difference between the sequence number announced by a SYNC packet and the expected sequence number.

If a reception failure occurs, the receiver schedules a timer with random value which is in the same range as the initial NACK timer. When the timer expires, the receiver sends a `SYNC_ERR` packet to report a serious sequence error if no such packets are detected. The receiver also does a reset and tries to synchronise with the current sequence number of the data stream. In addition, the receiver notifies the application that a reset has been performed.

At the sender side, upon reception of a `SYNC_ERR` packet, LRMP will notify the application of such an event together with the interval of lost sequence numbers. It is for the application to decide to take any remedial actions. For example it may ignore the event for a continuous data stream, or resend this part of the data according to the level of importance. This mechanism allows applications to control the level of quality of service and thus prevents the disturbance caused by a particular receiver with a very bad network connection. This is especially suitable for WAN based applications.

F.4.5 Flow Control

Unlike TCP which uses a window based flow control, LRMP uses a rate based flow control mechanism. That is, LRMP sends data packets, including repair packets, at a rate specified by an application. This rate is adapted dynamically to better suit available network bandwidth.

In order to avoid to aggravate network congestion, a simple and intuitive way is to keep transmission rate to the effective data rate, i.e., the rate with loss excluded. However this would not work in practice, because other applications using a best effort algorithm or with no adapted flow control will fill that bandwidth. The algorithm currently used in LRMP is based on loss statistics collected through RTCP RR's (receive reports). Only "bad" receivers affect the flow control. Transmission rate is decreased when there are many receivers with a significant loss rate, and it is increased when there are a few. The purpose of this algorithm is to keep an acceptable speed for the majority of receivers, and to try to satisfy a few worst receivers when possible. This algorithm may cause reception failure at the worst receivers, as discussed in the previous section.

F.4.6 Ordered Packet Delivery

While in-order packets are immediately delivered to the application, out-of-order packets are maintained in the cache by LRMP receivers in waiting repair packets. Once missing packets are received, cached packets are removed from the cache and delivered to the application. In addition, LRMP provides users with possibility to configure this option out, which packets could be delivered in reception order if the application desires. The reception cache has a limited size, depending on the used transmission rate, but should be large enough for loss repair. It is recommended that the buffer size should correspond to 10 seconds - 1 minutes of data packets at the full specified transmission rate (e.g., between 80 KB and 480 KB at 64 kb/s). However if there are many out-of-order packets and repairs do not arrive, the reception cache may become full. In this case, a reception failure event will be generated.

F.5 Data Packet Format

Data packets are in the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|reserved |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronisation source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                       application data                        |
|                            ....~~                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

V: version. It is set to the current RTP version number - 2.
P: padding. If it is set, the last byte of the packet contains the number of padding bytes. reserved: should be set to 0.
M: more data marker. It is set to 0 to indicate that more data will follow, it is set to 1 to indicate the boundary of a data segment, e.g. a video frame.
PT: payload type.
sequencenumber: it contains low 16 bits of the sequence number.
timestamp: the time associated to the following data. It is application dependent.
ssrc: a unique identifier of synchronisation source (data source).

F.6 Control Packet Formats

Five control packet types are specified in LRMP: `ECHO` and `ECHO_ACK` packets are used to measure the round trip time between two users; `NACK` packets are used to report packet loss; `SYNC` packets inform the receivers the outgoing sequence number reached; `SYNC_ERR` packets report an unrecoverable sequence error at a receiver. All these packets are implemented as RTCP control packets. Like RTCP, several LRMP packets can be multiplexed into one out-going packet.

F.6.1 Common Header

All LRMP control packets are started with the following common header:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|   ST    | PT=$RTCP_LRMP$|             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sender's SSRC                         |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

where:

V: version, the same as defined in the data packet format.
P: padding, the same as defined in the data packet format.
ST: subtype. It specifies the control packet type within LRMP.
PT: payload type. It is set to 206 for LRMP control packets.
length: the length of the packet in 32 bit word minus one.
SSRC: the identifier of the sender.

F.6.2 ECHO packet

When a participant becomes a sender in a group, other participants need to measure the round trip time to this participant. This is done by sending an ECHO packet. An ECHO packet can be sent together with RTCP RR packet. Before an `ECHO_ACK` packet is received from the sender, the round trip time is set by default to 1 second.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|  ST=0   | PT=$RTCP_LRMP$|             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sender's SSRC                         |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                             SSRC 1                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                         (next SSRC ...)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

ST: subtype. It is set to 0 for ECHO packets.
SSRC 1: the SSRC of the first destination. The value of 0xFFFFFFFF is given with a special meaning, it represents the wildcard address, i.e., the whole group.
timestamp: the time of sending this packet. It is the middle 32 bits of NTP time. So to query the round trip time to all participants, one can just set the destination ssrc to 0xFFFFFFFF. But this is rarely needed in LRMP.

F.6.3 ECHO ACK packet

A participant responds to an ECHO packet by sending an `ECHO_ACK` packet. `ECHO_ACK` packets can be sent together with `RTCP SR` packet.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|  ST=1   | PT=$RTCP_LRMP$|           length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sender's SSRC                         |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                            SSRC 1                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                delay since received ECHO packet               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                         (next SSRC ...)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

ST: subtype. It is set to 1 for `ECHO_ACK` packets.
SSRC 1: the SSRC of the originator of the first `ECHO` packet.
timestamp: the timestamp in the received `ECHO` packet.
delay: delay since received the ECHO packet. It is expressed in units of fraction of `1/65536` seconds.

F.6.4 SYNC packet

`SYNC` packets have two purposes:

- if issued by participants who are senders, they inform others the current sequence number reached;
- if issued by participants who never send data, they inform the start sequence number.

They are used specially to avoid missing starting and ending packets in a data stream. A `SYNC` packet can be sent together with `RTCP RR` packet if one is not a sender, or with `RTCP SR` if one is a sender. In addition, senders should send at least one SYNC packet when it becomes idle.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|   ST=2  | PT=$RTCP\_LRMP$|           length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sender's SSRC                         |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                             Seqno                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

Seqno: sequence number to be used in the next sending packet.

F.6.5 NACK packet

NACK packets report a set of lost sequence numbers and request the resend.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|   ST=3  | PT=RTCP_LRMP  |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sender's SSRC                         |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                             SSRC 1                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              FSL              |              BLP              |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                         (next SSRC ...)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

$SSRC1$: the SSRC of the first data source to which loss is detected.
FSL: first seqno lost.
BLP: bitmask of the following lost packets. A bit is set to 1 if the corresponding sequence number is lost. The value of $0x0101$ means that packets with seqno FSL, FSL + 1, FSL + 9 are lost.

F.6.6 SYNC ERROR packet

SYNC ERROR packets just report a reception failure event. No acknowledgement or action are required by default.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|   ST=4  | PT=RTCP_LRMP  |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sender's SSRC                         |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                             SSRC 1                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| CA  |           NL            |              FSL              |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                         (next SSRC ...)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

CA: cause. Value encodings are:
- 0 - cause unknown.
- 1 - buffer overrun.
- 2 - exceeded max number of repair tries.
- 3 - sender lost.
- 4 - sender left.
- +4 - not used.

NL: number of lost packets.
FSL: first seqno lost.

Appendix G

Synchronisation

G.1 Introduction

To stay synchronised, computers on a network exchange information about the time via a time service protocol and adjust their own clock accordingly. To move ahead they may add time to the clock, but moving backwards is more difficult. It is important that distributed clocks be monotonic (Lamport, 1978). Time cannot step backwards since this would confuse programs that use timestamps for determining precedence. To make backward adjustments, the clock rate is slowed down until the correct time is reached.

Most host clocks are set by eyeball-and-wristwatch to within a minute or two and rarely checked after that. Many of these are maintained by some sort of battery-backed clock/calender device using a room-temperature quartz oscillator that may drift seconds per day and can go for weeks between manual corrections. For many applications, especially those designed to operate in a distributed internet environment, much greater accuracy, stability and reliability are required.

G.1.1 The Network Time Protocol (NTP)

Network Time Protocol (NTP)(Mills, 1992) is used to synchronise the time of a computer client or server to another server or reference time source, such as a radio or satellite receiver or modem. It provides client accuracies typically within a millisecond on LANs and up to a few tens of milliseconds on WANs relative to a primary server synchronised to Coordinated Universal Time (UTC) via a Global Positioning Service (GPS) receiver, for example. Typical NTP configurations utilize multiple redundant servers and diverse network paths, in order to achieve high accuracy and reliability. In this thesis, a server of Aston University has been chosen as a reference time source for the experimental phase. A tool which explains in next subsection has used to synchronise the time between machines and server used for the experiment.

G.1.2 How to improve the synchronisation accuracy

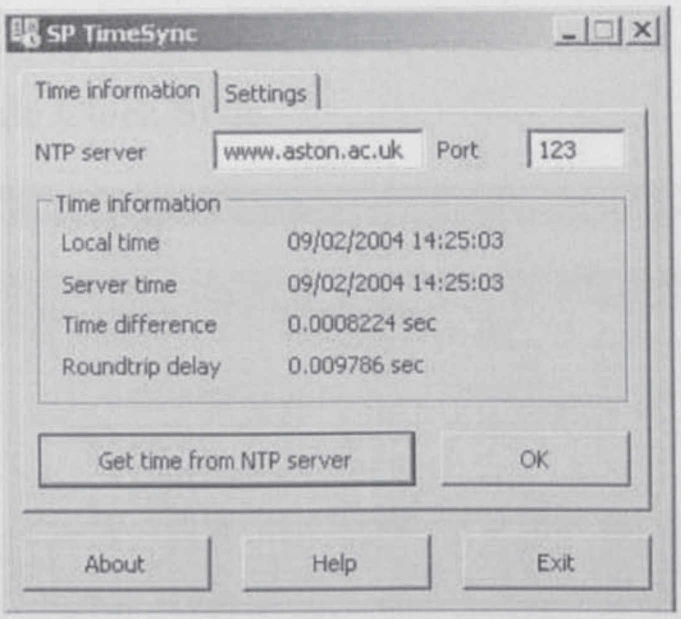
The accuracy of time synchronisation depends on the time required to transmit the data over the network (both from client to server and from server to client). It does not depend on the amount of time spent by server in order to process a request. The faster network used, the more accurate result can be theoretically achieve. There are several common steps which can help to minimise the roundtrip delay and improve the synchronisation accuracy:

- Choose the closest NTP server.
- Increase the speed of Internet connection.
- Don't use the Internet connection for other purposes during the time synchronisation. This will help to avoid additional delay caused by time multiplexing of the single Internet connection.

G.2 Synchronisation tools

This section gives a brief about tools used for synchronise the clock system. In this thesis, three different tools has been used and tested: Automic Time, SNTP, and SP TimeSync.

G.2.1 SP TimeSync


SP TimeSync

Time information | Settings

NTP server: www.aston.ac.uk Port: 123

Time information
- Local time: 09/02/2004 14:25:03
- Server time: 09/02/2004 14:25:03
- Time difference: 0.0008224 sec
- Roundtrip delay: 0.009786 sec

Figure G.1: Synchroniser Tool: SP TimeSync

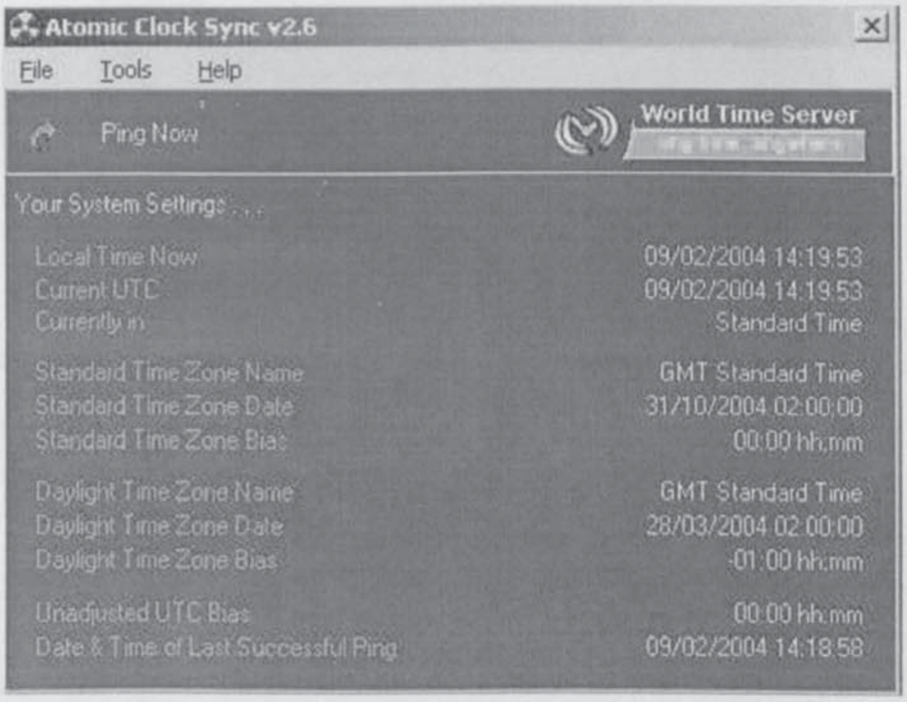
SP TimeSync is a software product which lets synchronise the computer's clock with any Internet atomic clock (time server). It uses a high-precision Network Time Protocol (NTP) which provides accuracy of several milliseconds depending on the characteristics of the synchronisation source and network paths. In this thesis, the software is used and it gives the accuracy of reading up to a milliseconds different. The accuracy of this software is up to 1 millisecond.

Operations

To retrieve the time from time server, the machine needs to have a connection to the Internet. When it requests a time from the time server, SP TimeSync will show the server time, the time difference, and the delay roundtrip. Roundtrip delay the amount of time spent to send a request to NTP server and to receive the reply from it. This delay is caused by signal propagation over the Internet. The identity of a reference server can be specified in a NTP server field in the software which is used as a high-precision time source. To achieve the maximum precision on a machine, the chosen

NTP server must be situated the closest to the machine in order to reduce interferences from the network.

G.2.2 Atomic Clock Sync



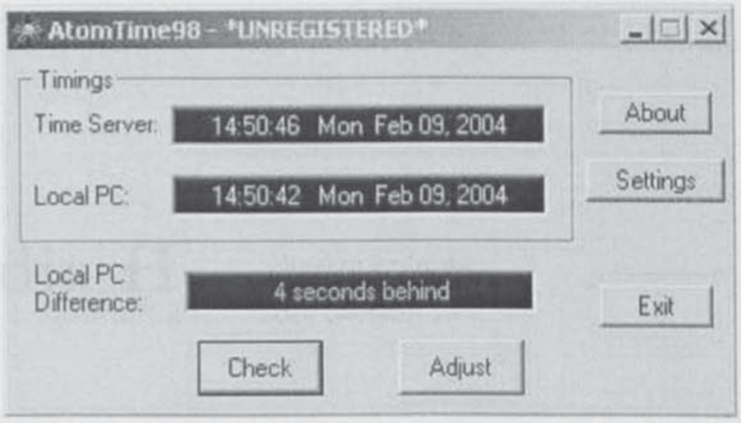
```
Your System Settings . . .

  Local Time Now                          09/02/2004 14:19:53
  Current UTC                             09/02/2004 14:19:53
  Currently in                                  Standard Time

  Standard Time Zone Name                   GMT Standard Time
  Standard Time Zone Date                 31/10/2004 02:00:00
  Standard Time Zone Bias                        00:00 hh:mm

  Daylight Time Zone Name                   GMT Standard Time
  Daylight Time Zone Date                 28/03/2004 02:00:00
  Daylight Time Zone Bias                       -01:00 hh:mm

  Unadjusted UTC Bias                            00:00 hh:mm
  Date & Time of Last Successful Ping     09/02/2004 14:18:58
```

Figure G.2: Synchroniser Tool: Atomic Clock Sync

This software connects to one of the time servers operated by the US National Institute of Standards and Technology (NIST), and will set the time of the clock in the our computer to the atomic time-scale operated by NIST. The advantage is the user does not need to memorise the time server address (IP). The accuracy of this software is 1 second.

G.2.3 AtomTime98

AtomTime is a Windows application which will connect to the Atomic Clock time server in Boulder, Colorado (USA) via the Internet and fetch the current atomic clock time value. This software also allows to set with different reference time server. It compares this value with PC time and displays the difference. For further information



Time Server: 14:50:46 Mon Feb 09, 2004

Local PC: 14:50:42 Mon Feb 09, 2004

Local PC Difference: 4 seconds behind

Figure G.3: Synchroniser Tool: AtomTime98

on AtomTime98, please visit this link `http://www.atomtime.com/manual.html`. The accuracy of this software is 1 second.

Appendix H

Reliability

This appendix explains how reliability of a system can be measured.

H.1 How to measure reliability

There are a number of ways to measure reliability for a system:

- One way to define reliability is the probability of failure-free operation of a computer program for a specified time in a specified environment (Musa *et al.*, 1990). For example, if we have a reliability of 0.99 for 10 hours the program will, in average, during these 10 hours fail one time out of hundred. Another way to express reliability is failure intensity, i.e. failures per time unit.

- For hardware reliability Mean Time To Failure, MTTF, is often used. It is also used in software reliability but more and more seldom. The reason for not to use it, is that there are many cases in software reliability in which MTTF is undefined. Even if the relation between reliability and failure intensity depends on the model that we used, they usually vary during the test phase, as faults are removed.

- By checking the reachability of a host to different host/server. In a case of of multicasting, check the reachability of different hosts and routers via IP multicast (Dressler, 2002). For IP Unicast users, ICMP messages are used to proved the connectivity of different IP end systems. However, there is no such tool for IP multicast connections. Dressler (2002) has proposed MRM to provide a tool to measure the reliability for multicasting. Dressler (2002) has differentiated the measurement of reliability and QoS. In QoS measurement, packet loss ratio, absolute delay, and jitter(variation of the delay) are considered.

Appendix I

User Manual for WebCOM System

I.1 Introduction

This appendix explains the manual of the WebCOM System.

I.2 Software Requirement

Since the WebCOM System was developed as a platform independent Java applet application, there is no requirements for any specific operating system on which this application can be run. A Web browser enabled for Java applets requires JRE, JDK or JSE to be installed to the system. The Java installation can be downloaded from Sun web sites at `http://java.sun.com/`.

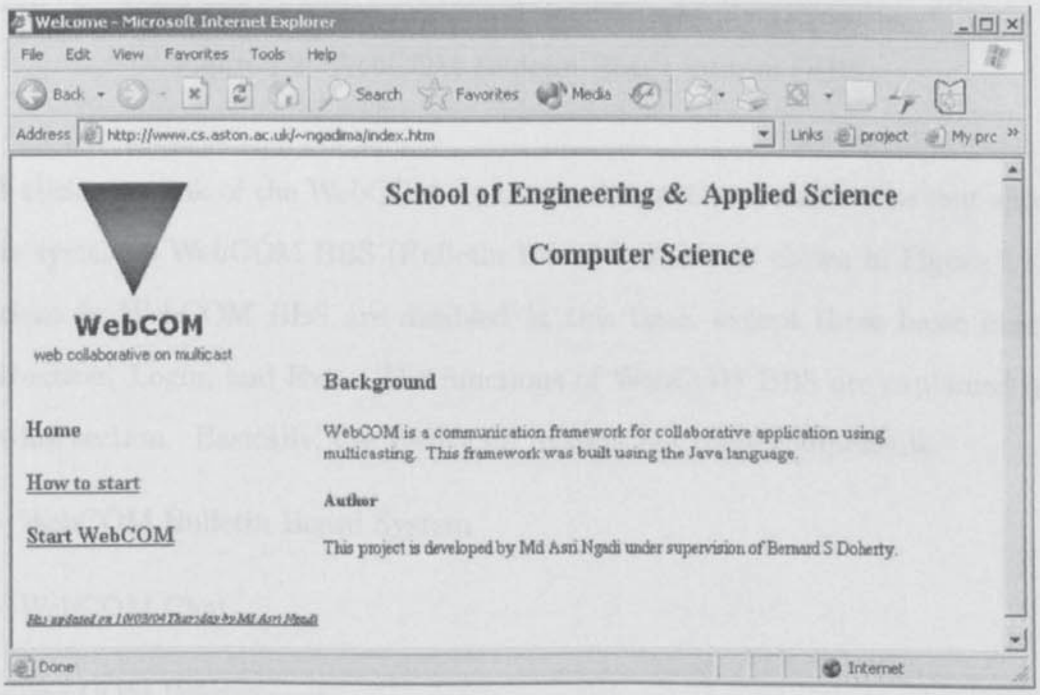
I.3 Hardware Requirement

There is no special requirements for hardware because the WebCOM System works on any machine as long as there is a network connection and the machine supports a Web browser.

I.4 Setting up the Web browser

The Web browser needs to be configured so it can run Java applets in the browser. The Java Run-time Environment (JRE)[1] is required to set up the Web browser to enable Java-based applications or applets. The next configuration is to enable multicast socket creation by adding commands in the Java policy file. This file consists of a set of rules on how the browser should act.

I.5 Accessing the system


Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.cs.aston.ac.uk/~ngadima/index.htm
Links project My prc
Done
Internet

School of Engineering & Applied Science

Computer Science

WebCOM

web collaborative on multicast

Home

How to start

Start WebCOM

Background

WebCOM is a communication framework for collaborative application using multicasting. This framework was built using the Java language.

Author

This project is developed by Md Asri Ngadi under supervision of Bernard S Doherty.

Has updated on 10/03/04 Thursday by Md Asri Ngadi

Figure I.1: The Main Page of WebCOM System

A network connection is required to access the system. The WebCOM System can be accessed using a Web browser by entering a URL address. The first page, as shown in Figure I.1, appears when accessing the main page of WebCOM System. This page provides an introduction to the system, guidelines to use the system, and a link to the WebCOM System.

[1]obtain from `http://java.sun.com/`

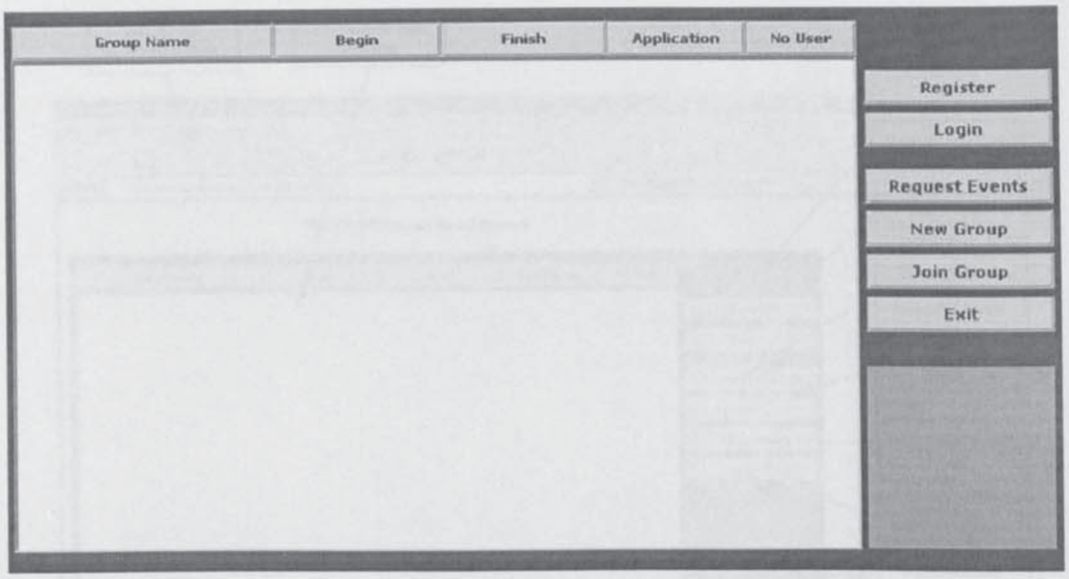


Figure I.2: WebCOM Bulletin Board System (BBS)

A click on a link of the WebCOM System will start the system. The first entrance to the system is WebCOM BBS (Bulletin Board System) as shown in Figure I.2. All functions in WebCOM BBS are disabled at this time, except three basic functions i.e. Register, Login, and Exit. The functions of WebCOM BBS are explained in the following section. Basically, the WebCOM System has three components:

- WebCOM Bulletin Board System
- WebCOM Chat
- WebCOM Whiteboard

I.6 WebCOM Bulletin Board System

The WebCOM BBS provides six functions as shows in Figure I.3:

- Register a new user.
  This option is used to register a new user into the WebCOM System.
- Login into WebCOM System.
  This option is used to login into the WebCOM System to enable other functions

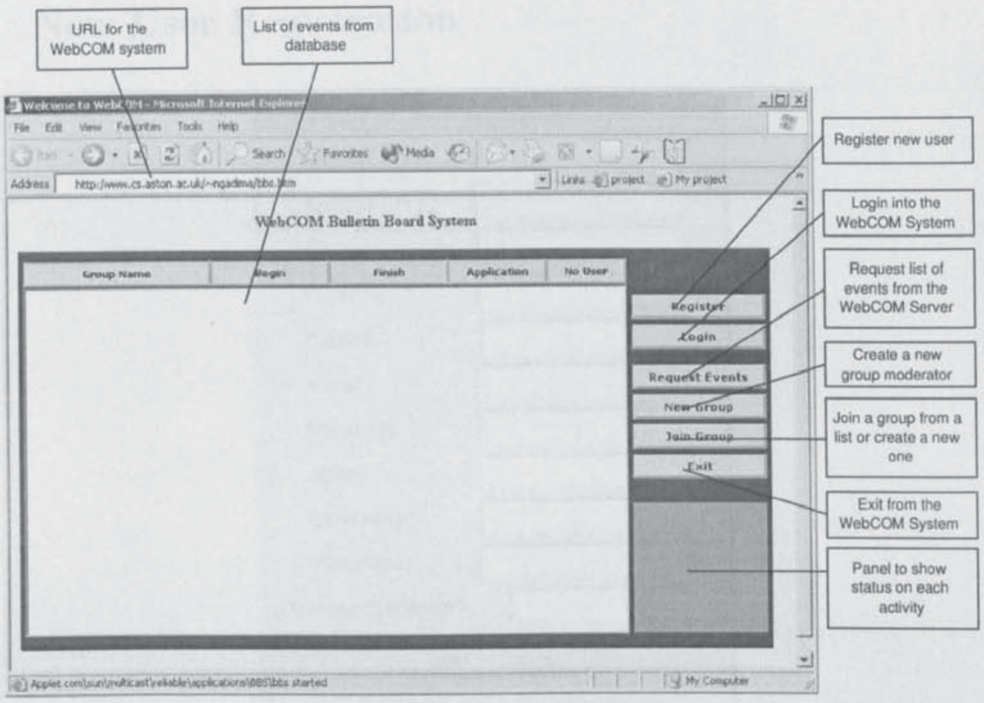

URL for the WebCOM system
List of events from database
Welcome to WebCOM - Microsoft Internet Explorer
http://www.cs.aston.ac.uk/~ngadima/bbs.htm
WebCOM Bulletin Board System
Group Name
Begin
Finish
Application
No User
Register
Login
Request Events
New Group
Join Group
Exit
Register new user
Login into the WebCOM System
Request list of events from the WebCOM Server
Create a new group moderator
Join a group from a list or create a new one
Exit from the WebCOM System
Panel to show status on each activity
Applet com\sun\multicast\reliable\applications\BBS\bbs started
My Computer

Figure I.3: WebCOM BBS

on WebCOM BBS.

- Request a list of event in the WebCOM System.

  This option is used to load events from WebCOM Server.

- Create a new group in the WebCOM System.

  This option is used to create a new group for WebCOM System.

- Join existing group.

  This option is used to join any group available inside the WebCOM System or a valid user-defined group.

- Exit from the WebCOM System.

  This option is used to exit from the WebCOM System.

As mentioned earlier, only Register, Login, and Exit functions are enabled on first entry into the WebCOM System. Other functions are enabled if the login process is successful. So, any user who intends to use the WebCOM System needs to register on the WebCOM System.

I.6.1 New User Registration

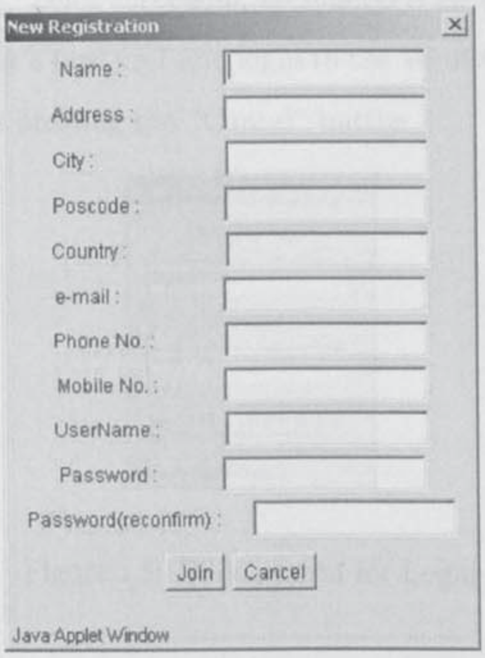
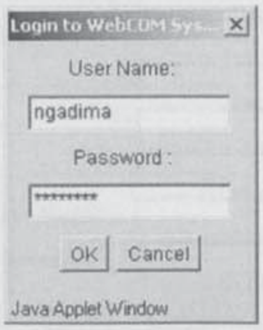


Figure I.4: Form Dialog for Creating a New User

If a user is not yet a member of the WebCOM System, he/she can choose the "Register" button to register their details. Figure I.4 shows an example of the form used to register the details of a new user. Complete the form correctly and remember the username and password that you chose. Once finished, press the "Create" button to send details to the WebCOM Server. If the user already exists in the system, an error message appears on the "BBS Status" otherwise a successful message appears on the "BBS Status". Cancelation of the registration process is by pressing the "Cancel" button.

I.6.2 Login

If a user is already a member for the system, he/she can login to the WebCOM System by pressing the "Login" button. Once the "Login" button is pressed, a dialog message as shown in Figure I.5 comes up, requesting a username and a password. Enter username and password used in the new user registration process. Once completed, press the "OK" button. The logon status can be found on the "BBS Status" area on the WebCOM BBS interface. Once successful, all functions on the WebCOM System are enabled except registering a new user and login to the WebCOM System. Cancelation of the login process is by pressing the "Cancel" button.



User Name: ngadima

Password : *********

Figure I.5: Dialog Form for Login

I.6.3 Events Request

If the logon process is successful, a user can request a list of events from WebCOM Server by pressing the "Request Events" button on the WebCOM BBS as shown in Figure I.3. If any event exists in database, the WebCOM Server will multicast the events to all participants of WebCOM BBS, which will appear on the list area of WebCOM BBS. Otherwise, an error indicating no event available in the database appears on the "BBS Status" area.

I.6.4 New Group Creation

If there is no group available in the WebCOM System, a user that has a privilege as moderator (assigned by the administrator of the Web server), can create a new group. A moderator can create a new group by pressing "New Group" button on the WebCOM BBS. A dialog form for creating a new group as shown in Figure I.6 appears after pressing the "New Group" button. The moderator needs to complete the form as directed by the Web server administrator. The Web server administrator will provide a multicast address (Class D) and port number for use by a new group.

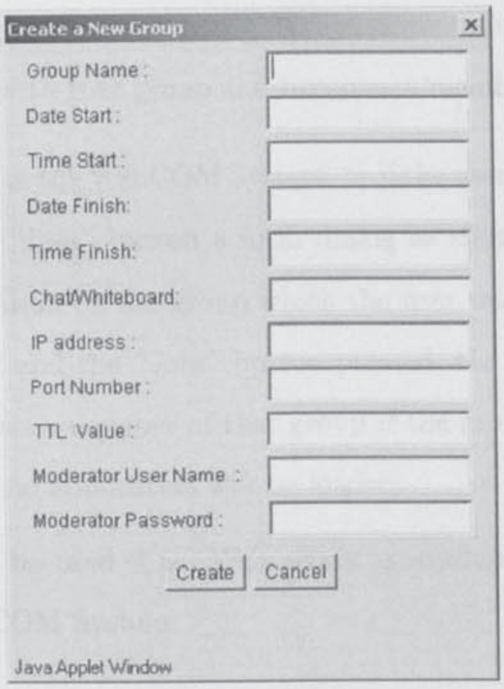


Figure I.6: Dialog Form for Creating a New Group

I.6.5 Group Participation

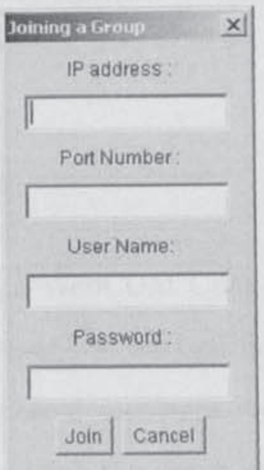


Figure I.7: Dialog Form for Joining a Group

A group can be joined in two different ways:

- If there is an event or events in the WebCOM System, one is selected from the list. After pressing the "Join" button, the user automatically loads an application that is associated with that group and becomes a member of that group as well.

- If there is no event in the WebCOM System, or if the user does not select an event, after pressing the "Join" button a form dialog as shown in Figure I.7 appears requesting information on the group which the user intended to join. Once the form is completed and the "Join" button pressed, the selected application will appear and becomes a member of that group if the group information is correct and valid. If not, no application will be loaded.

  This method can be used if no Web server is available at the initial stage of running the WebCOM System.

I.6.6 Exit

The user can exit from the system by pressing the "Exit" button on the WebCOM BBS user interface. Once the button is pressed, the current window will close and be disconnected from the WebCOM System. A leaving message will appear on other participant's application if any.

I.7 WebCOM Chat

When you choose a group using a WebCOM Chat a window as shown in Figure I.8 appears. The user can enter a message on the text field provided and press the "Enter" button once completed. All received messages are viewed on the WebCOM Chat window. Every time a new user joins the group or leaves the group, a "joining" or "leaving" message appears in the WebCOM Chat window. Pressing the "Exit" button will close the WebCOM Chat window and the user is disconnected from the group.

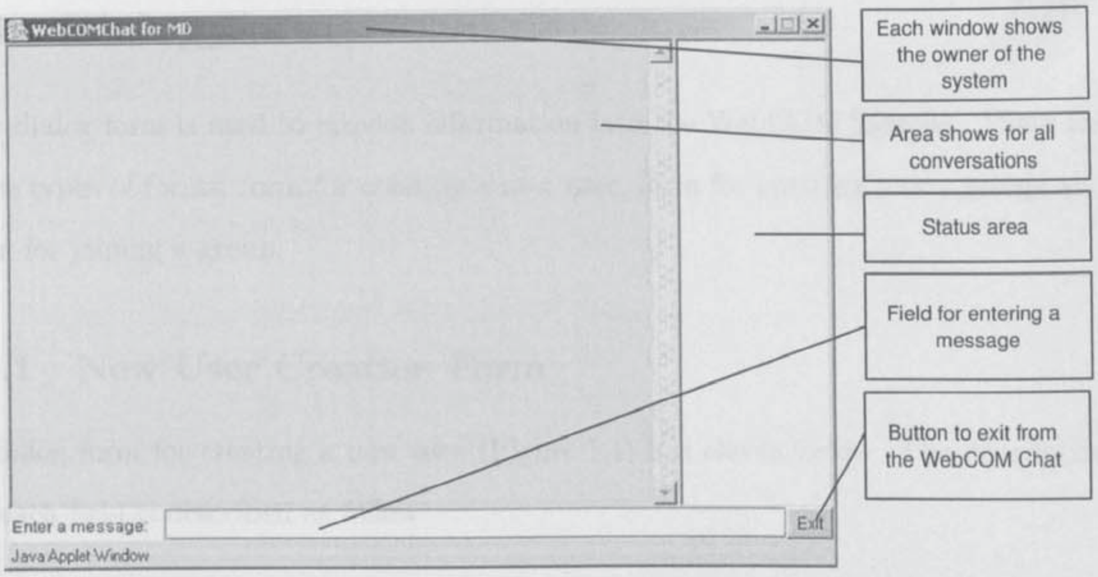

WebCOMChat for MD
Enter a message:
Exit
Java Applet Window
Each window shows the owner of the system
Area shows for all conversations
Status area
Field for entering a message
Button to exit from the WebCOM Chat

Figure I.8: WebCOM Chat

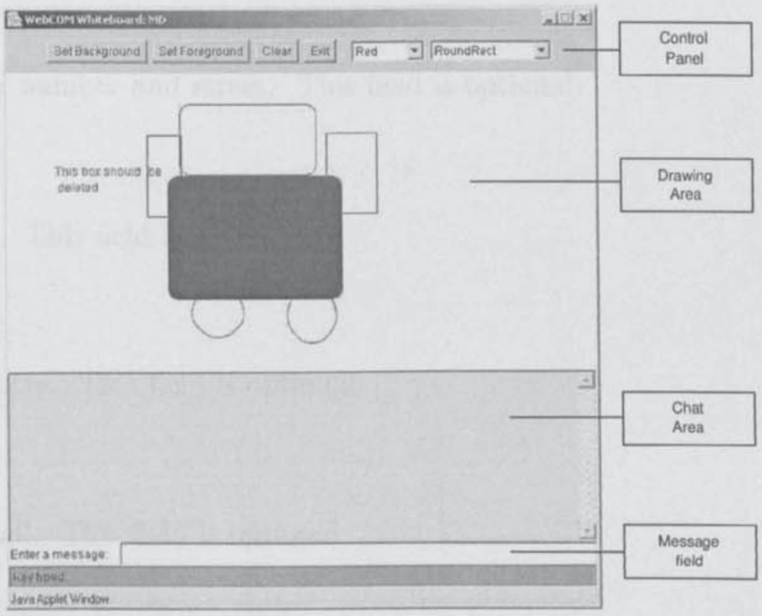


Figure I.9: WebCOM Whiteboard

I.8 WebCOM Whiteboard

When a user choose a group using a WebCOM Whiteboard, a window as shown in Figure I.9 appears. The WebCOM Whiteboard provides basic drawing, typing, and chat facilities as well.

I.9 Dialog Form

The dialog form is used to provide information into the WebCOM System. There are three types of forms: form for creating a new user, form for creating a new group, and form for joining a group.

I.9.1 New User Creation Form

A dialog form for creating a new user (Figure I.4) has eleven fields. The description for each field is described as follow:

- Name.

  Full user name. This field is compulsory.

- Address.

  Flat/House number and street. This field is optional.

- City.

  User's city. This field is optional.

- Country.

  User's country. This field is optional.

- E-mail.

  User's e-mail. This field is optional.

- Phone Number.

  User's phone number. This field is optional.

- Mobile Number.

  User's mobile phone number. This field is optional.

- UserName.

  User's username. This field is compulsory.

- Password.

  User's password. This field is compulsory.

- Password (reconfirm).

  User's password. This field is compulsory.

I.9.2 New Group Creation Form

A dialog form for creating a new group (Figure I.6) has ten fields. The description for each field is described as follow:

- Group Name.

  Name of a new group. The name should represent the activity of the group. The name of the group should not too shot or too long. The appropriate length is within 5 to 15 characters. This field is compulsory.

- Date Start.

  Date when the group will start.. Enter in free format. Time format that normally used is DD/MM/YYYY. This field is compulsory.

- Time Start.

  Time when the group will start. Enter in free format. Normally format 24 hours (HH:MM) is used. This field is compulsory.

- Date Finish.

  Date when the group will finish. Enter in free format . Time format that normally used is DD/MM/YYYY. This field is compulsory.

- Time Finish.

  Time when the group will finish. Enter in free format. Normally format 24 hours (HH:MM) is used. This field is compulsory.

- Chat/Whiteboard.

  Type of application that will be used by the group. Enter exactly the word "Chat" or "Whiteboard" only for this field. This field is compulsory.

- IP address.

  IP address used for the group. Enter a valid address which supplied from your network/server administrator. Contact your network administrator to get the address. This field is compulsory. Don't enter an address without asking your network administrator. If the WebCOM System is used use an internal private network, the user can enter any valid multicast address.

- Port Number.

  Port number for the group. Enter a valid number which supplied by your network/server administrator. Contact your network administrator to get a port number. This field is compulsory. Don't enter a port number without asking your network administrator. If the WebCOM System is used for an internal private network, the user can enter any valid port number.

- TTL Value.

  TTL[2] value for the group. Enter a valid value for TTL. The value of TTL is used to cover the number of hops likely to be needed, which increases with geographic distance. This field is compulsory.

- Moderator Username.

  Username for moderator. Only someone with a valid moderator name can create a new group. This field is compulsory.

- Moderator Password.

  Password for moderator. Only someone with a valid moderator password can create a new group. This field is compulsory.

[2]The multicast Time-To-Live (TTL) value specifies the number of routers (hops) that multicast traffic is permitted to pass through before expiring on the network. For each router (hop), the original specified TTL is decremented by one (1). When its TTL reaches a value of zero (0), each multicast datagram expires and is no longer forwarded through the network to other subnets.

I.9.3 Group Participation Form

A dialog form for creating a new group (Figure I.5) has ten fields. The description for each field is described as follow:

- Group Name.

  Name of a new group. The name should represent the activity of the group. The name of the group should not too shot or too long. The appropriate length is within 5 to 15 characters. This field is compulsory.

- Chat/Whiteboard.

  Type of the application that will use by the group. Enter exactly the word "Chat" or "Whiteboard" only for this field. This field is compulsory.

- IP address.

  IP address used for the group. Enter a valid address which supplied from your network/server administrator. Contact your network administrator to get the address. This field is compulsory. Don't enter an address without asking your network administrator. If the WebCOM System is used use an internal private network, the user can enter any valid multicast address.

- Port Number.

  Port number for the group. Enter a valid number which supplied by your network/server administrator. Contact your network administrator to get a port number. This field is compulsory. Don't enter a port number without asking your network administrator. If the WebCOM System is used for an internal private network, the user can enter any valid port number.

- UserName.

  User's username. This field is compulsory.

- Password.

  User's password. This field is compulsory.

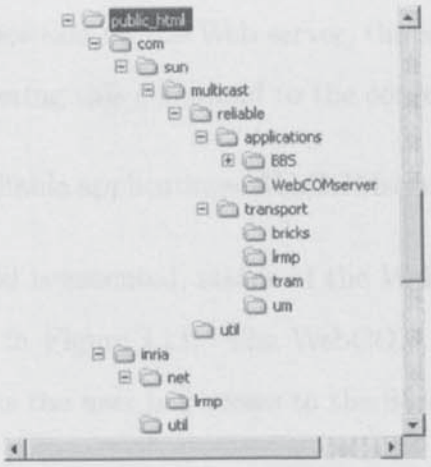
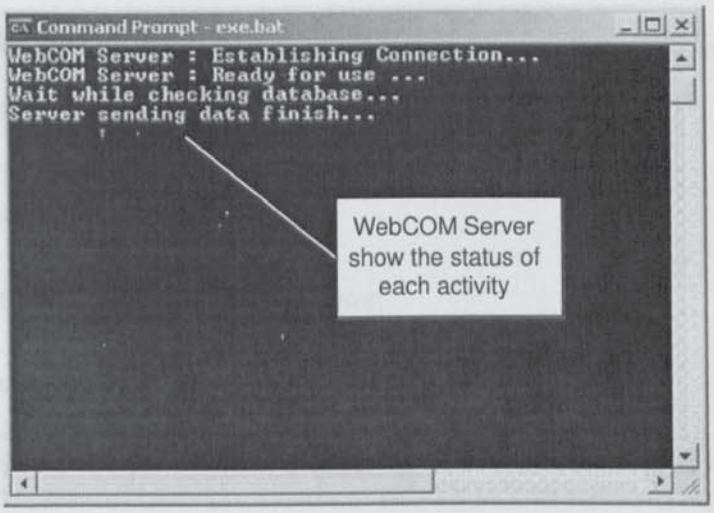


Figure I.10: Directories Arrangement for WebCOM Server



```
WebCOM Server : Establishing Connection...
WebCOM Server : Ready for use ...
Wait while checking database...
Server sending data finish...
```

Figure I.11: WebCOM Server

I.10 Administrator

This part is used for the administrator of the Web server.

I.10.1 WebCOM Server

The WebCOM Server is located inside the Web server. The directory of the application must be placed as shown in Figure I.10. The LRMP class should be downloaded and place in this directory.

Once the files are in the position on the Web server, the administrator can run the WebCOM application by entering this command to the console:

```
java com.sun.multicast.reliable.applications.WebCOMserver.WebCOMserver
```

When the above command is executed, status of the WebCOM Server appears on the console screen as shown in Figure I.11. The WebCOM Server can execute from any host and server as long as the user has access to the files.

I.10.2 Database

All information is store in the database located on the server. There are database for user information, group information, moderator information and server information. The arrangement of databases are:

Format of Database

The format used in each database is as follows:

- Format for user information:

Name	Address	City	Postcode	Country	Email	PhoneNo	MobileNo	UserName	Password	Password2

- Format for group information:

GroupName	StartDate	StartTime	EndDate	EndTime	Application Type	IP	Port Number	TTL	UserName	Password	NoUser

- Format for moderator information:

userName	password	ipAddress	portNo

- Format for server information:

ipAddress	portNo	TTL

Appendix J

Programming Listing

J.1 WebCOM Server

```
/*
*   Md Asri Ngadi (c) 2004
*   WebCOM Server
*   Create a user interface for WebCOM System allow user to login to the WebCOM System,
*   register a user into the system, create a new group for moderator (please request one
*   from administrator), request events happening in the system, join any group on the
*   list or a specified known group.
*
*   Most of the coding is from my self and some of them has obtained from various sources
*   on the Internet and modified according to the objective of the research.
*
*
*/

package com.sun.multicast.reliable.applications.WebCOMserver;

import javax.swing.*;
import java.io.*;
import java.net.*;
import java.awt.event.*;
import java.awt.*;

import java.lang.String;
import com.sun.multicast.util.*;
import com.sun.multicast.reliable.*;
import com.sun.multicast.reliable.transport.*;
import com.sun.multicast.reliable.transport.lrmp.*;
import com.sun.multicast.reliable.transport.lrmp.LRMPPacketSocket;

public class WebCOMserver {
    private InetAddress ia;
    private int port;
    private String portInitial,ipInitial,ttlInitial;



    Thread thread              = null;
    BufferedReader  console   = null;
    PrintStream streamOut = null;

    ReceiverThread receiver = null;
    RMPacketSocket socket = null;
    String transportName = "LRMP";


    WebCOMserver(){

        /*
        *   Check information on database to initialise
        *   WebCOM Server. IP address, port and ttl value
        *   are information in the database.
        */
        try{
            FileReader serverFile = new FileReader("server.txt");
            BufferedReader initialState = new BufferedReader(serverFile);
            boolean eofserverFile = false;

            String line = initialState.readLine();
            StringBuffer lineBfr = new StringBuffer(line);

            while (!eofserverFile){
                int j = lineBfr.indexOf(";");
                ipInitial = lineBfr.substring(0,j);
                lineBfr = lineBfr.delete(0,j+1);

                j = lineBfr.indexOf(";");
                portInitial = lineBfr.substring(0,j);
                lineBfr = lineBfr.delete(0,j+1);

                j = lineBfr.indexOf(";");
                ttlInitial = lineBfr.substring(0,j);
                line = initialState.readLine();
                if (line == null)eofserverFile = true;

            }
            initialState.close();
        }catch(Exception ex){System.out.println(ex.toString());}

        port = Integer.parseInt(portInitial);
        System.out.println("WebCOM Server : Establishing Connection... ");
        try{
                ia = InetAddress.getByName(ipInitial);
        }catch (UnknownHostException e){
                throw new IllegalArgumentException();
        }

        try{
            createSocket();
        } catch (Exception ex){
            System.out.println("Error: Creating the socket.");
        }
        System.out.println("WebCOM Server : Ready for use ... ");
        sendSignIn();

    }

    /**
     * Inner class to receive messages and display them
     */
    public class ReceiverThread extends Thread
    {
        private boolean done = false;
        private boolean check_strRequestList, check_strNewGroup, check_strNewUser, check_strLogin = false;

        String messageList;
        byte[] dataList, data2, data3, data4;
        DatagramPacket dpList, dp2, dp3, dp4;

        //Standard WORD control used
        String strRequestList    = "LIST0000000000000000";
        String strNewGroup       = "NEWGROUP000000000000";
        String strNewUser        = "NEWUSER0000000000000";
        String strLogin          = "LOGIN000000000000000";

        public void run()
        {
            while (!done)
            {
                try
                {
                    /*
                    *   Receive information from client
                    */
                    DatagramPacket dp_receive = socket.receive();
                    String message = new String(dp_receive.getData(), "UTF8");

                    StringBuffer message_bfr = new StringBuffer(message);


                    String strCheck = message_bfr.substring(0,20);
                    message_bfr = message_bfr.delete(0,21);

                    /*
                    *   Filter and check all information from client
                    */
                    check_strRequestList = strRequestList.equals(strCheck);
                    check_strNewGroup = strNewGroup.equals(strCheck);
                    check_strNewUser = strNewUser.equals(strCheck);
                    check_strLogin = strLogin.equals(strCheck);

                    /*
                    *   If request is event listing
                    */
                    if ( check_strRequestList == true )
                    {
                        System.out.println("Wait while checking database... ");
                        try{
                            FileReader inFileList = new FileReader("groupInfo.txt");
                            BufferedReader groupInfo = new BufferedReader(inFileList);
                            boolean eofInFileList = false;

                            String line = groupInfo.readLine();

                            while (!eofInFileList){
                                if (line!=null){
```

```
                //send all record to user
                try
                {
                    messageList = "ACKLISTSTART00000000;"+line;
                    dataList = messageList.getBytes("UTF8");
                    dpList = new DatagramPacket(dataList, dataList.length, ia, port);
                    socket.send(dpList);
                    //System.out.println("Server sending data...");
                    Thread.sleep(200);
                } catch (Exception ex) {
                    System.out.println("Error3: SendMessage - actionPerformed()");
                }
                line = groupInfo.readLine();

            }else if (line == null){
                eofInFileList = true;
                try
                {
                    messageList = "ACKLISTEND0000000000;";
                    dataList = messageList.getBytes("UTF8");
                    dpList = new DatagramPacket(dataList, dataList.length, ia, port);
                    socket.send(dpList);
                    System.out.println("Server sending data finish...");
                    check_strRequestList=false;
                    Thread.sleep(200);
                } catch (Exception ex) {
                    System.out.println("Error3: SendMessage - actionPerformed()");
                }

            }

        }

        groupInfo.close();

    } catch (IOException e){
        System.out.println(e.toString());
    }


}else if (check_strNewGroup == true){
    /*
    *   If request is new group creation
    */
    boolean statusModerator =false;
    boolean statusGroupInFile =true;

    int j = 0;
    System.out.println("Creating New Group...");

    //check moderator status
    FileReader inFileModerator = new FileReader("moderatorInfo.txt");
    BufferedReader moderatorInfo = new BufferedReader(inFileModerator);
    boolean eofmoderator = false;

    FileReader inFileGroup = new FileReader("groupInfo.txt");
    BufferedReader GroupInfo = new BufferedReader(inFileGroup);
    boolean eofGroup = false;

    /*
    *   read the file
    */
    String readFile = moderatorInfo.readLine();
    String readFileGroup = GroupInfo.readLine();


    /*
    *   put the received information  into string buffer
    */
    String buffString = message_bfr.substring(0,message_bfr.length());
    StringBuffer strCheckModerator = new StringBuffer(buffString);



    for(int i = 1 ; i <= 6; i++)
    {
        j = strCheckModerator.indexOf(";");
        strCheckModerator = strCheckModerator.delete(0,j+1);
        //System.out.println("Contents of message = "+ strCheckModerator);
    }
    j = strCheckModerator.indexOf(";");
    String ipGroupReceived = strCheckModerator.substring(0,j);
    strCheckModerator = strCheckModerator.delete(0,j+1);

    j = strCheckModerator.indexOf(";");
    String portGroupReceived = strCheckModerator.substring(0,j);
    strCheckModerator = strCheckModerator.delete(0,j+1);

    j = strCheckModerator.indexOf(";");
    strCheckModerator = strCheckModerator.delete(0,j+1);

    j = strCheckModerator.indexOf(";");
    String usernameModeratorReceived = strCheckModerator.substring(0,j);
    strCheckModerator = strCheckModerator.delete(0,j+1);

    j = strCheckModerator.indexOf(";");
```

```
String passwordModeratorReceived = strCheckModerator.substring(0,j);

/*
*   compare received information with information inside database
*/
if (readFile == null ){
    eofmoderator = true;
}else {
    while (!eofmoderator){
        StringBuffer strDatabaseCheckModerator = new StringBuffer(readFile);
        j = strDatabaseCheckModerator.indexOf(";");
        String usernameModeratorInDB = strDatabaseCheckModerator.substring(0,j);
        strDatabaseCheckModerator = strDatabaseCheckModerator.delete(0,j+1);
        j = strDatabaseCheckModerator.indexOf(";");
        String passwordModeratorInDB = strDatabaseCheckModerator.substring(0,j);

        boolean check_loginameInFileModerator = usernameModeratorReceived.equals(usernameModeratorInDB);
        boolean check_passwordInFileModerator = passwordModeratorReceived.equals(passwordModeratorInDB);

        if (check_loginameInFileModerator == true &&
        check_passwordInFileModerator == true){
            statusModerator = true;
            eofmoderator = true;
        }

        readFile = moderatorInfo.readLine();
        if (readFile == null ){
            eofmoderator = true;
        }
    }
}

/*
*   Check the existence of group in database
*/
if (readFileGroup == null ){
    eofGroup = true;
}else {
    while (!eofGroup){

        StringBuffer strDatabaseCheckGroup = new StringBuffer(readFileGroup);

        for(int i = 1 ; i <= 6; i++)
        {
            j = strDatabaseCheckGroup.indexOf(";");
            strDatabaseCheckGroup = strDatabaseCheckGroup.delete(0,j+1);

        }
        j = strDatabaseCheckGroup.indexOf(";");
        String ipGroupInDatabase = strDatabaseCheckGroup.substring(0,j);
        strDatabaseCheckGroup = strDatabaseCheckGroup.delete(0,j+1);

        j = strDatabaseCheckGroup.indexOf(";");
        String portGroupInDatabase = strDatabaseCheckGroup.substring(0,j);
        strDatabaseCheckGroup = strDatabaseCheckGroup.delete(0,j+1);

        j = strDatabaseCheckGroup.indexOf(";");
        strDatabaseCheckGroup = strDatabaseCheckGroup.delete(0,j+1);

        j = strDatabaseCheckGroup.indexOf(";");
        String usernameModeratorInDB = strDatabaseCheckGroup.substring(0,j);
        strDatabaseCheckGroup = strDatabaseCheckGroup.delete(0,j+1);

        j = strDatabaseCheckGroup.indexOf(";");
        String passwordModeratorInDB = strDatabaseCheckGroup.substring(0,j);

        boolean check_ipGroup  = ipGroupReceived.equals(ipGroupInDatabase);
        boolean check_portGroup= portGroupReceived.equals(portGroupInDatabase);
        boolean check_loginame = usernameModeratorReceived.equals(usernameModeratorInDB);
        boolean check_password = passwordModeratorReceived.equals(passwordModeratorInDB);

        if (check_ipGroup == true && check_portGroup == true ){
            statusGroupInFile = true;
            eofGroup = true;
        }

        readFileGroup = GroupInfo.readLine();
        if (readFileGroup == null ){
            eofGroup= true;
        }
    }
}

moderatorInfo.close();
GroupInfo.close();

/*if authorised and no same group available,
* a new group info is update into the database
*/
if (statusModerator == true && statusGroupInFile == false){
    FileWriter inFile = new FileWriter("groupInfo.txt",true);//append
    BufferedWriter userInfo = new BufferedWriter(inFile);
    message_bfr = message_bfr.append("0;");
    String msg = message_bfr.substring(0,message_bfr.length());
    userInfo.write(msg);
    System.out.println("Data saved into database = "+msg);
```

```
        userInfo.write("\n");
        userInfo.close();


    }else
    {//send error message to the client
        System.out.println("Error: Data unsaved into database.");
    }



}else if (check_strNewUser == true){
    System.out.println("Creating New User...");
    FileWriter inFile = new FileWriter("userInfo.txt",true);//append
    BufferedWriter userInfo = new BufferedWriter(inFile);
    String msg = message_bfr.substring(0,message_bfr.length());
    userInfo.write(msg);
    userInfo.write("\n");
    userInfo.close();

}else if (check_strLogin = true){
    boolean statusUser = false;
    System.out.println("Check Username and Password...");

    FileReader inFileUser = new FileReader("userInfo.txt");
    BufferedReader userInfo = new BufferedReader(inFileUser);
    boolean eofuser = false;

    String readFile = userInfo.readLine();
    System.out.println("Contents of database = "+ readFile);

    //take the username and password of moderator that receive
    String buffString = message_bfr.substring(0,message_bfr.length());
    StringBuffer strCheckUser = new StringBuffer(buffString);
    //System.out.println("Message from client = "+ strCheckModerator);

    int j = strCheckUser.indexOf(";");
    String usernameUser = strCheckUser.substring(0,j);
    String nameUserInDB="";
    strCheckUser = strCheckUser.delete(0,j+1);
    j = strCheckUser.indexOf(";");
    String passwordUser = strCheckUser.substring(0,j);
    System.out.println("username and password receive = "+ usernameUser + " & " + passwordUser);
    //System.out.println("password moderator receive = "+ passwordModerator);

    //compare with database
    if (readFile == null ){
        eofuser = true;
    }else {
        while ( !eofuser ){
            StringBuffer strDatabaseCheckUser = new StringBuffer(readFile);

            j = strDatabaseCheckUser.indexOf(";");
            nameUserInDB = strDatabaseCheckUser.substring(0,j);
            strDatabaseCheckUser = strDatabaseCheckUser.delete(0,j+1);

            for(int i = 2 ; i <= 8; i++)
            {
                j = strDatabaseCheckUser.indexOf(";");
                strDatabaseCheckUser = strDatabaseCheckUser.delete(0,j+1);

            }


            j = strDatabaseCheckUser.indexOf(";");
            String usernameUserInDB = strDatabaseCheckUser.substring(0,j);
            strDatabaseCheckUser = strDatabaseCheckUser.delete(0,j+1);
            j = strDatabaseCheckUser.indexOf(";");
            String passwordUserInDB = strDatabaseCheckUser.substring(0,j);
            System.out.println("username and password in database = "+ usernameUserInDB + " & " + passwordUserInDB);

            boolean check_loginame = usernameUser.equals(usernameUserInDB);
            boolean check_password = passwordUser.equals(passwordUserInDB);

            //System.out.println("username moderator DB = " + usernameModeratorInDB);
            //System.out.println("password moderator DB = " + passwordModeratorInDB);
            //System.out.println("Status username moderator = " + check_loginame);
            //System.out.println("Status password moderator = " + check_password);
            System.out.println("Status of user = "+ check_loginame + " & " + check_password);

            if (check_loginame == true && check_password == true){
                statusUser = true;
                eofuser = true;
            }

            readFile = userInfo.readLine();
            if (readFile == null ){
                eofuser = true;
            }
        }
        userInfo.close();

        //if users are registered ACK POS else ACK NEG with their name and password as well
        if (statusUser == true){
            try
                {
                    messageList = "ACKLOGINOK0000000000;" + usernameUser + ";" + passwordUser + ";" + nameUserInDB + ";";
```

```
                                    dataList = messageList.getBytes("UTF8");
                                    dpList = new DatagramPacket(dataList, dataList.length, ia, port);
                                    socket.send(dpList);
                                    System.out.println("Server sending ACKLOGIN Successfull.");
                                    check_strRequestList=false;
                                    Thread.sleep(300);
                                } catch (Exception ex) {
                                    System.out.println(ex.toString());
                                }


                        }else
                        {//send error message to the client
                            try
                            {
                                messageList = "ACKLOGINK00000000000;" + usernameUser + ";" + passwordUser + ";" + nameUserInDB + ";";
                                dataList = messageList.getBytes("UTF8");
                                dpList = new DatagramPacket(dataList, dataList.length, ia, port);
                                socket.send(dpList);
                                System.out.println("Server sending ACKLOGIN Fail.");
                                check_strRequestList=false;
                                Thread.sleep(200);
                            } catch (Exception ex) {
                                System.out.println(ex.toString());
                            }
                        }

                    }
                }else {
                    System.out.println("Data receive : "+ message);
                }

            } catch (Exception ex) {
                System.out.println("Client has left a WebCOM System.");
            }
        }
    }

    public void terminate() {
        done = true;
        sendSignoff();
    }
}

/**
*  Creating socket for WebCOM Server
*/
public void createSocket() throws IOException, RMException
{
    TransportProfile tp = null;

    if (transportName.equals("LRMP")) {
        LRMPTransportProfile lrmptp = null;
        try {
            lrmptp = (LRMPTransportProfile) new LRMPTransportProfile(ia, port);
        } catch (InvalidMulticastAddressException e) {
            throw new ImpossibleException(e);
        }
        lrmptp.setTTL((byte) Integer.parseInt(ttlInitial));
        lrmptp.setOrdered(true);
        tp = lrmptp;
    } else {
        throw new AssertFailedException();
    }

    try {
        socket = tp.createRMPacketSocket(TransportProfile.SEND_RECEIVE);
    } catch (InvalidTransportProfileException ie) {
        throw new ImpossibleException(ie);
    } catch (UnsupportedException ue) {
        throw new ImpossibleException(ue);
    } catch (IOException ioe){
    } catch (RMException rme){
    } catch (Exception tk){System.out.println(tk.toString());}

    receiver = new ReceiverThread();
    receiver.start();
}

//close all component of WebCOM Server
public void close(){
    if (receiver != null) {
        receiver.terminate();
    }
    if (socket != null) {
        sendSignoff();
        socket.close();
    }

}

    /**
  * Sends a  signoff message when window is closing
  */
public void sendSignoff() {
    String message = "WEBCOMSERVEROFF00000;Server is off.";
```

```
        try {
            byte data[] = message.getBytes("UTF8");
            DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
            socket.send(dp);
        } catch (Exception ex) {}
        close();
    }

    /**
    *   Sends a signIn message to other users when window is closing
    */
    private void sendSignIn() {
        String message = "WEBCOMSERVERREADY000; Server is ready.";

        try {
            byte data[] = message.getBytes("UTF8");
            DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
            socket.send(dp);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    /*
    *   Main method
    */
    public static void main(String args[])
    {
        try{
             WebCOMserver server = new WebCOMserver();
        }catch (Exception  e){
            System.out.println(e.toString());
        }

     }
}
```

J.2 WebCOM Bulletin Board System

```
/*
*   Md Asri Ngadi 2004
*   WebCOM Client - BBS System
*   Create a user interface for WebCOM System allow user to login to the WebCOM System,
*   register a user into the system, create a new group for moderator (please request one
*   from administrator), request events happening in the system, join any group on the
*   list or a specified known group.
*
*   Most of the coding is from the author.  Some of the coding has obtained from various sources
*   on the Internet and modified according to the objective of the research.
*
*
*/

package com.sun.multicast.reliable.applications.BBS;

import javax.swing.*;
import java.io.*;
import java.net.*;
import java.net.URL;
import java.awt.*;
import java.awt.event.*;
import netscape.javascript.*;
import javax.swing.text.InternationalFormatter;
import java.lang.String;

import com.sun.multicast.util.*;
import com.sun.multicast.reliable.*;
import com.sun.multicast.reliable.transport.*;
import com.sun.multicast.reliable.transport.lrmp.*;
import com.sun.multicast.reliable.transport.lrmp.LRMPPacketSocket;


public class bbs extends JApplet implements ActionListener,Runnable
{
    //variables for creating user interface
    private JPanel panel = new JPanel();
    private List list = new List(10);
    private JTextField textfield = new JTextField();
    private JButton button = new JButton();
    private Font ver = new Font("Verdana",1,11);
    private Font ver2 = new Font("Verdana",1,10);

    private JButton membership = new JButton();
    private JButton logon = new JButton();
    private JButton requestEvents = new JButton();
    private JButton createNewGroup = new JButton();
    private JButton joinGroup = new JButton();
    private JButton exit = new JButton();
    private JTextArea statusBBS = new JTextArea("",5,1);
    private JTextArea propertyBBS = new JTextArea("",5,1);

    private JButton nameGroup = new JButton();
    private JButton dateBegin = new JButton();
    private JButton dateEnd = new JButton();
    private JButton infoGroup = new JButton();
    private JButton  noUsers = new JButton();

    private boolean done = false;
    private boolean verbose = false;
    private boolean statusRegisteredMembership = false;

    //variables for creating socket enable communications
    private ReceiverThread receiver = null;
    private RMPacketSocket socket = null;
    private String transportName = "LRMP";
    private int port;
    private InetAddress ia;
    private String ipJoin, portJoin, applicationJoin, ttlJoin;
    private String ipServer,portServer, ttlServer;


    //variables for storing the data receiving from WebCOM server
    private String tableInfoGroup[][]= new String[99][99];
    private int row =0;
    private int col = 0;

    // variables for holding the types of applications and
    // local username and password
    private String applicationTypeChat = "Chat";
    private String applicationTypeWB = "Whiteboard";
    private String LoginNameSystem = "";
    private String PasswordSystem = "";
    private String UserNameSystem = "";
    private String prefix;

    public void init()
    {
        /*
        *  creating user interfaces
        */
        getContentPane().add(panel);
        panel.setLayout(null);
        panel.setBackground(new Color(16,48,156)); //biru tua

        panel.add(membership);
        membership.setBounds(620,45,135,25);
        membership.setFont(ver);
        membership.setText("Register");
        membership.addActionListener(this);


        panel.add(logon);
        logon.setBounds(620,75,135,25);
        logon.setFont(ver);
        logon.setText("Login");
        logon.addActionListener(this);


        panel.add(requestEvents);
        requestEvents.setBounds(620,115,135,25);
        requestEvents.setFont(ver);
        requestEvents.setText("Request Events");
        requestEvents.addActionListener(this);

        panel.add(createNewGroup);
        createNewGroup.setBounds(620,145,135,25);
        createNewGroup.setFont(ver);
        createNewGroup.setText("New Group");
        createNewGroup.addActionListener(this);

        panel.add(joinGroup);
        joinGroup.setBounds(620,175,135,25);
        joinGroup.setFont(ver);
        joinGroup.setText("Join Group");
        joinGroup.addActionListener(this);

        panel.add(exit);
        exit.setBounds(620,205,135,25);
        exit.setFont(ver);
        exit.setText("Exit");
        exit.addActionListener(this);


        panel.add(statusBBS);
        statusBBS.setEditable(false);
        statusBBS.setLayout(null);
        statusBBS.setBackground(Color.gray);
        statusBBS.setBounds(620,255,135,130);
        statusBBS.invalidate();
        statusBBS.setLineWrap(true);
        statusBBS.setWrapStyleWord(true);

        panel.add(propertyBBS);
        propertyBBS.setEditable(false);
        propertyBBS.setLayout(null);
        propertyBBS.setBackground(Color.gray);
        propertyBBS.setBounds(620,15,135,25);
        propertyBBS.invalidate();
        propertyBBS.setFont(ver);
        propertyBBS.setLineWrap(true);
        propertyBBS.setWrapStyleWord(true);

        panel.add(nameGroup);
        nameGroup.setBounds(5,10,190,25);
        nameGroup.setFont(ver2);
        nameGroup.setText("Group Name");
        panel.add(list);
        list.setBounds(5,36,610,350);
        list.setFont(ver2);
        list.addActionListener(this);

        panel.add(dateBegin);
        dateBegin.setBounds(195,10,120,25);
        dateBegin.setFont(ver2);
        dateBegin.setText("Begin");

        panel.add(dateEnd);
        dateEnd.setBounds(315,10,120,25);
        dateEnd.setFont(ver2);
        dateEnd.setText("Finish");

        panel.add(infoGroup);
        infoGroup.setBounds(435,10,100,25);
        infoGroup.setFont(ver2);
        infoGroup.setText("Application");

        panel.add(noUsers);
        noUsers.setBounds(535,10,80,25);
        noUsers.setFont(ver2);
        noUsers.setText("No User");


        /*
        *   register each BBS to WebCOM server
        */
        portServer = getParameter("portServer");
        ipServer = getParameter("ipServer");
        ttlServer = getParameter("ttlServer");
```

```

      System.out.println("ipServer: "+ipServer + "  portServer: "+portServer);


      port = Integer.parseInt(portServer);
      try{
          ia = InetAddress.getByName(ipServer);
      }catch (UnknownHostException e){
          throw new IllegalArgumentException();
      }

      try{
          createSocket();
      } catch (Exception ex){
          System.out.println("Error: Creating the socket.");
      }
}

  /*
   *   Create a socket
  */
  private void createSocket() throws IOException, RMException
  {
      TransportProfile tp = null;
      if (transportName.equals("LRMP")) {
          LRMPTransportProfile lrmptp = null;
          try {
              lrmptp = (LRMPTransportProfile) new LRMPTransportProfile(ia, port);
          } catch (InvalidMulticastAddressException e) {
              throw new ImpossibleException(e);
          }
          lrmptp.setTTL((byte) Integer.parseInt(ttlServer));
          lrmptp.setOrdered(true);
          tp = lrmptp;
      } else {
          throw new AssertFailedException();
      }

      try {
          socket = tp.createRMPacketSocket(TransportProfile.SEND_RECEIVE);
      } catch (InvalidTransportProfileException ie) {
          throw new ImpossibleException(ie);
      } catch (UnsupportedException ue) {
          throw new ImpossibleException(ue);
      } catch (IOException ioe){
      } catch (RMException rme){
      } catch (Exception tk){}

      receiver = new ReceiverThread();
      receiver.start();
  }

  /**
   * Inner class to receive data / control
  */
  private class ReceiverThread extends Thread
  {
      private boolean done = false;

      /*
       *  Every incoming data will be filtered according
       *  to WORD control.
       *   ACKLISTSTART - ack from WebCOM Server indicate the
       *                  beginning list of events
       *   ACKLISTEND - ack from WebCOM Server indicate the
       *                transmission of events was finished
       *   ACKLOGINOK - ack from WebCOM Server indicate the
       *                login process was successful
       *   ACKLOGINKO - ack from WebCOM Server indicate the
       *                login process was unsuccessful
      */

      private boolean fromServer = false;
      private boolean fromClient = false;
      private boolean chkListFrServerStart = false;
      private boolean chkListFrServerFinish = false;
      private boolean chkMembershipOK = false;
      private boolean chkMembershipKO= false;
      private boolean isNewList = true;
      private boolean chkUser = false;
      private boolean registered = false;

      //check incoming data is ACK on list of events
      private String strListFrServerStart   = "ACKLISTSTART00000000";
      private String strListFrServerFinish  = "ACKLISTEND0000000000";

      //check incoming login was successfull or not
      private String strMembershipOK        = "ACKLOGINOK0000000000";
      private String strMembershipKO        = "ACKLOGINKO0000000000";

      private String message;

      public void run()
      {
          while (!done)
          {
                try
                {
                    //data receiving
                    DatagramPacket dp = socket.receive();
                    message = new String(dp.getData(), "UTF8");

                    StringBuffer msg_bfr = new StringBuffer(message);

                    //separating the WORD control
                    String chk_status = msg_bfr.substring(0,20);
                    msg_bfr = msg_bfr.delete(0,21);

                    //compare the WORD control
                    chkListFrServerStart = strListFrServerStart.equals(chk_status);
                    chkListFrServerFinish =  strListFrServerFinish.equals(chk_status);
                    chkMembershipOK = strMembershipOK.equals(chk_status);
                    chkMembershipKO = strMembershipKO.equals(chk_status);

                    //check the status of valid user
                    if (chkMembershipOK == true && registered == false){
                        int z = msg_bfr.indexOf(";");
                        String str_usernameFrServer = msg_bfr.substring(0,z);
                        msg_bfr = msg_bfr.delete(0,z+1);
                        z = msg_bfr.indexOf(";");
                        String str_passwordFrServer = msg_bfr.substring(0,z);
                        msg_bfr = msg_bfr.delete(0,z+1);
                        z = msg_bfr.indexOf(";");
                        String str_nameUserFrServer = msg_bfr.substring(0,z);
                        msg_bfr = msg_bfr.delete(0,z+1);
                        UserNameSystem = str_nameUserFrServer;


                        boolean chkStatusLogin = LoginNameSystem.equals(str_usernameFrServer);
                        boolean chkStatusPassword =  PasswordSystem.equals(str_passwordFrServer);


                        if (chkStatusLogin == true && chkStatusPassword == true){

                            statusRegisteredMembership = true;
                            statusBBS.insert("Login was Successfull.\n\n",0);
                            registered = true;
                            propertyBBS.setText(UserNameSystem);
                        }
                        else
                            statusRegisteredMembership = false;
                    }else if (chkMembershipKO == true){
                        statusBBS.insert("Login was Failed.\n\n",0);
                    }



                    //if list from server is true
                    if (chkListFrServerFinish == true)
                    {   statusBBS.insert("Receiving finished.\n\n",0);
                        isNewList = true;
                        chkListFrServerStart = false;
                    }

                    if (chkListFrServerStart==true)
                    {
                        //remove view of list each time receive from WebCOM server
                        if (isNewList == true)
                        {
                            list.removeAll();
                            isNewList = false;
                            for (int e=0; e<=98; e++){
                                for (int f=0; f<=98; f++){
                                    tableInfoGroup[e][f] = null;
                                }
                            }
                            row = 0;
                            col = 0;
                            statusBBS.insert("Receiving started.\n\n",0);
                        }

                        //indicate position of ";"
                        int j = msg_bfr.indexOf(";");

                        /*
                        *   Arrange all incoming data into the a viewable list.
                        *   Store all incoming data into multi-array
                        */
                        String messageView = msg_bfr.substring(0,j);
                        tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//groupname
                        msg_bfr = msg_bfr.delete(0,j+1);
                        StringBuffer strView = new StringBuffer();

                        for (int i=0;i<=4;i++) strView = strView.append(" ");

                        strView = strView.append(messageView);

                        j = msg_bfr.indexOf(";");
                        messageView = msg_bfr.substring(0,j);
                        tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//begin date

                        msg_bfr = msg_bfr.delete(0,j+1);
                    for (int i=strView.length();i<=66-strView.length();i++) strView = strView.append("_");
                    strView = strView.append(messageView);

                    j = msg_bfr.indexOf(";");
                    messageView = msg_bfr.substring(0,j);
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//begin time

                    msg_bfr = msg_bfr.delete(0,j+1);
                    for (int i=strView.length();i<=93-strView.length();i++) strView = strView.append("_");
                    strView = strView.append(messageView);

                    j = msg_bfr.indexOf(";");
                    messageView = msg_bfr.substring(0,j);
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//finish date

                    msg_bfr = msg_bfr.delete(0,j+1);
                    for (int i=strView.length();i<=110-strView.length();i++) strView = strView.append("_");
                    strView = strView.append(messageView);

                    j = msg_bfr.indexOf(";");
                    messageView = msg_bfr.substring(0,j);
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//finish time

                    msg_bfr = msg_bfr.delete(0,j+1);
                    for (int i=strView.length();i<=135-strView.length();i++) strView = strView.append("_");
                    strView = strView.append(messageView);

                    j = msg_bfr.indexOf(";");
                    messageView = msg_bfr.substring(0,j);
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//application
                    msg_bfr = msg_bfr.delete(0,j+1);

                    for (int i=strView.length();i<=155-strView.length();i++) strView = strView.append("_");
                    strView = strView.append(messageView);


                    j = msg_bfr.indexOf(";");
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//ipaddress
                    msg_bfr = msg_bfr.delete(0,j+1);

                    j = msg_bfr.indexOf(";");
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//port
                    msg_bfr = msg_bfr.delete(0,j+1);

                     j = msg_bfr.indexOf(";");
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//TTL
                    msg_bfr = msg_bfr.delete(0,j+1);

                    j = msg_bfr.indexOf(";");
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//user name
                    msg_bfr = msg_bfr.delete(0,j+1);

                    j = msg_bfr.indexOf(";");
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//password
                    msg_bfr = msg_bfr.delete(0,j+1);

                    j = msg_bfr.indexOf(";");
                    messageView = msg_bfr.substring(0,j);
                    tableInfoGroup[col++][row] = msg_bfr.substring(0,j);//number of user
                    msg_bfr = msg_bfr.delete(0,j+1);

                    for (int i=strView.length();i<=195-strView.length();i++) strView = strView.append("_");
                    strView = strView.append(messageView);

                    String strList = strView.substring(0,strView.length());
                    list.add(strList);
                    strList = null;

                    row++;
                    col = 0;

                }//if

            } catch (Exception ex) {System.out.println(ex.toString());}
        }//while
    }

    public void terminate() {
        done = true;
    }
}


/*
*   Response on user's selection when pressing a button on the BBS
*/

public void actionPerformed(ActionEvent e)
{   String message;
    int indexSelected = 0;
    String textSelected = null;

    boolean statusList = false;
    boolean statusListSelect = false;

```

```

        //create a new user
        if (e.getSource() == membership && statusRegisteredMembership == false){

            //statusBBS.insert("Creating New User.\n\n",0);
            createNewMember createNonMembership = new createNewMember (new Frame(""));
            requestFocus();
            createNonMembership.dispose();
        }

        //login into the system if user already registered
        if (e.getSource() == logon && statusRegisteredMembership == false){
            statusBBS.insert("Login to the system. \n\n",0);
            Login login = new Login (new Frame(""));

            requestFocus();
            login.dispose();
        }

        //send request for current event available to WebCOM server
        if (e.getSource() == requestEvents)
        {
            statusBBS.setLineWrap(true);
            statusBBS.insert("Incoming Data. Please WAIT... \n\n",0);

            try{
                list.removeAll();
            }catch (Exception ee){}

            message = "LIST0000000000000000;";
            try
            {
                byte[] data = message.getBytes("UTF8");
                DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
                socket.send(dp);
            } catch (Exception ex) {
                System.out.println("Error: "+ ex.toString());
                statusBBS.insert("Error while sending message to the server. Please contact your admin.\n\n",0);
            }

        }

        //create a new group into the system
        if (e.getSource() == createNewGroup){
            if (statusRegisteredMembership == false){
                statusBBS.insert("ONLY accessible by moderator.
                Moderator needs to login as a user first before
                creating a new group.  You can request one from your admin.\n\n",0);
            }
            if (statusRegisteredMembership == true){
                //statusBBS.insert("Creating a new group.\n\n",0);
                CreateNewGroup createNewGroupButton = new CreateNewGroup (new Frame(""));
                requestFocus();
                createNewGroupButton.dispose();
                statusBBS.insert("Please check your group by pressing 'Request Events' button.\n\n",0);
            }
        }

        //join selection group from the list or known group
        if (e.getSource() == joinGroup ){
            if (statusRegisteredMembership == false){
                statusBBS.insert("ONLY accessible by a registered user. Please login to the system first.\n\n",0);
            } else if ( statusRegisteredMembership == true){
                //statusBBS.insert("Loading a system.  Please WAIT...\n\n",0);
                indexSelected = list.getSelectedIndex();
                textSelected = list.getSelectedItem();
            }

            if (textSelected != null && indexSelected >= 0){
                statusListSelect = true;
                statusList = true;
            } else {
                //no list available or no list selected
                statusListSelect = false;
                statusList = false;
            }

            if (statusList == true  &&
                statusListSelect == false &&
                statusRegisteredMembership == true)
            {
                JoinGroup joinButton = new JoinGroup (new Frame(""));
                requestFocus();
                joinButton.dispose();
            } else if (  statusList == false &&
                statusListSelect == false &&
                statusRegisteredMembership == true)
            {
                JoinGroup joinButton = new JoinGroup (new Frame(""));
                requestFocus();
                joinButton.dispose();
            } else if (   statusList == true &&
                statusListSelect == true &&
                statusRegisteredMembership == true)
            {
                //send request to use the application
                applicationJoin = tableInfoGroup[5][indexSelected];
                ipJoin = tableInfoGroup[6][indexSelected];
                portJoin = tableInfoGroup[7][indexSelected];
                ttlJoin = tableInfoGroup[8][indexSelected];

                boolean chkApplicationTypeChat =  applicationJoin.equals(applicationTypeChat);
                boolean chkApplicationTypeWB =  applicationJoin.equals(applicationTypeWB);

                if (chkApplicationTypeChat == true){
                    statusBBS.insert("Loading a system.  Please WAIT...\n\n",0);

                    try {
                        WebCOMChat chatUser = new WebCOMChat(ipJoin,portJoin,ttlJoin);
                    }catch (Exception ex){ex.toString();}
                } else if (chkApplicationTypeWB == true){
                    statusBBS.insert("Loading a system.  Please WAIT...\n\n",0);

                    try{
                        WebCOMWhiteboard whiteboardUser = new WebCOMWhiteboard(ipJoin,portJoin,ttlJoin);
                        //WebCOMWhiteboard whiteboardUser = new WebCOMWhiteboard();
                        //whiteboardUser.init();

                    }catch (Exception ex){ex.toString();}
                }

            }
        }


        //join the selection group in the list when double click
        if (e.getSource() == list)
        {
            //check either member or not
            if (statusRegisteredMembership == true){

                indexSelected = list.getSelectedIndex();
                textSelected = list.getSelectedItem();

                applicationJoin = tableInfoGroup[5][indexSelected];
                ipJoin = tableInfoGroup[6][indexSelected];
                portJoin = tableInfoGroup[7][indexSelected];
                ttlJoin = tableInfoGroup[8][indexSelected];

                boolean chkApplicationTypeChat =  applicationJoin.equals(applicationTypeChat);
                boolean chkApplicationTypeWB =  applicationJoin.equals(applicationTypeWB);


                if (chkApplicationTypeChat == true){
                    statusBBS.insert("Loading a system.  Please WAIT...\n\n",0);

                    try{
                        WebCOMChat WCChat = new WebCOMChat(ipJoin,portJoin,ttlJoin);
                    }catch(Exception ex){ex.toString();}

                } else if (chkApplicationTypeWB == true){
                    statusBBS.insert("Loading a system.  Please WAIT...\n\n",0);
                    try{
                        WebCOMWhiteboard whiteboardUser = new WebCOMWhiteboard(ipJoin,portJoin,ttlJoin);
                        //WebCOMWhiteboard whiteboardUser = new WebCOMWhiteboard();
                    }catch(Exception ex){ex.toString();}
                }
            }else if (statusRegisteredMembership == false){
                statusBBS.insert("Please login to the system first.\n\n",0);
            }
        }

        //exit from BBS
        if (e.getSource() == exit){
            close();
        }
    }

    /*
    * Create a Login dialog to the user and validate with WebCOM Server
    */
    private class Login extends Dialog implements ActionListener {
        private Button ok,can;
        private TextField loginName;
        private TextField password;

        Login(Frame frame){
            super(frame, "Login to WebCOM System", true);
            setLayout(new FlowLayout());

            loginName = new TextField(15);
            password = new TextField(15);
            password.setEchoChar('*');

            add( new Label("User Name:"));
            add(loginName);

            add( new Label("Password :"));
            add(password);
            addWindowListener(new WL());


            addOKNewCancelPanel();
            createFrame();
            //pack();
            //setVisible(true);
        }

        void addOKNewCancelPanel() {
            Panel p = new Panel();
            p.setLayout(new FlowLayout());
            createButtons( p );
            add( p );
        }

        void createButtons(Panel p) {
            p.add(ok = new Button("OK"));
            ok.addActionListener(this);

            p.add(can = new Button("Cancel"));
            can.addActionListener(this);
        }

        void createFrame() {
            Dimension d = getToolkit().getScreenSize();
            setLocation(d.width/2,d.height/3);
            setResizable(false);
            setSize(160,200);
            show();
        }


        public void actionPerformed(ActionEvent ae){
            if(ae.getSource() == ok) {
                if (loginName.getText() != null && password.getText() != null){
                    statusBBS.insert("Login to the System. Please Wait..\n\n",0);
                    String info = loginName.getText() +";"+ password.getText()+";";
                    LoginNameSystem = loginName.getText();
                    PasswordSystem = password.getText();
                    String message = "LOGIN000000000000000;"+info;
                    try
                    {
                        byte[] data = message.getBytes("UTF8");
                        DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
                        socket.send(dp);
                    } catch (Exception ex) {
                        System.out.println("Error: SendMessage - actionPerformed()");
                    }
                    setVisible(false);
                }else {
                    statusBBS.insert("Please check each entry in the form.\n\n",0);
                }

            }

            if(ae.getSource() == can) {
                setVisible(false);
            }
        }
        private class WL extends WindowAdapter {
            // If window closes, exit
            public void windowClosing(WindowEvent e) {
                setVisible(false);
            }
        }
    }


    /*
    *   Create a dialog of registration a New Member to the system
    *   and send to WebCOM Server for storing into database
    */

    private class createNewMember extends Dialog implements ActionListener
    {
        private Button ok,can;
        private TextField nameUser;
        private TextField address;
        private TextField city;
        private TextField postcode;
        private TextField country;
        private TextField e_mail;
        private TextField lineNo;
        private TextField mobileNo;
        private TextField userName;
        private TextField password1;
        private TextField password2;

        private Label nameUser1;
        private Label address1;
        private Label city1;
        private Label postcode1;
        private Label country1;
        private Label e_mail1;
        private Label lineNo1;
        private Label mobileNo1;
        private Label userName1;
        private Label password11;
```

```
     private Label password21;



     createNewMember(Frame frame){
         super(frame, "New Registration", true);
         setLayout(new FlowLayout());


         nameUser = new TextField(15);
         address = new TextField(15);
         city = new TextField(15);
         postcode = new TextField(15);
         country = new TextField(15);
         e_mail = new TextField(15);
         lineNo = new TextField(15);
         mobileNo = new TextField(15);
         userName = new TextField(15);
         password1 = new TextField(15);
         password1.setEchoChar('*');
         password2 = new TextField(15);
         password2.setEchoChar('*');

         nameUser1 =  new Label("Name :                 ");
         address1 =   new Label("Address :             ");
         city1 =      new Label("City :                      ");
         postcode1 =  new Label("Postcode :          ");
         country1 =   new Label("Country :               ");
         e_mail1 =    new Label("E-mail :                ");
         lineNo1 =    new Label("Phone No. :        ");
         mobileNo1 =  new Label("Mobile No. :       ");
         userName1 =  new Label("UserName :       ");
         password11 = new Label("Password :       ");
         password21 = new Label("Password(reconfirm) : ");

         nameUser1.setAlignment(1);
         add(nameUser1);
         add(nameUser);
         add(address1);
         add(address);
         add(city1);
         add(city);
         add(postcode1);
         add(postcode);
         add(country1);
         add(country);
         add(e_mail1);
         add(e_mail);
         add(lineNo1);
         add(lineNo);
         add(mobileNo1);
         add(mobileNo);
         add(userName1);
         add(userName);
         add(password11);
         add(password1);
         add(password21);
         add(password2);
         addWindowListener(new WL());
         addOKCancelPanel();
         createFrame();

     }

     void addOKCancelPanel() {
         Panel p = new Panel();
         p.setLayout(new FlowLayout());
         createButtons( p );
         add( p );
     }

     void createButtons(Panel p) {
         p.add(ok = new Button("Join"));
         ok.addActionListener(this);
         p.add(can = new Button("Cancel"));
         can.addActionListener(this);
     }

     void createFrame() {
         Dimension d = getToolkit().getScreenSize();
         setLocation(d.width/2,d.height/3);
         setResizable(false);
         setSize(300,400);
         show();
     }


     public void actionPerformed(ActionEvent ae){

         if(ae.getSource() == ok) {
             if (nameUser.getText()!=null && userName.getText()!= null &&password1.getText()!=null){
                 statusBBS.insert("Creating New User. Please Wait...\n\n",0);

                 String info = nameUser.getText() +";"+
                             address.getText()+";"+ city.getText()+";"+postcode.getText()
                             +";"+country.getText()+";"+e_mail.getText()+";"+lineNo.getText()+";"+
                                mobileNo.getText()+";"+userName.getText()+";"+password1.getText()+";";
                    String message = "NEWUSER0000000000000;"+info;

                    try
                    {
                        byte[] data = message.getBytes("UTF8");
                        DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
                        socket.send(dp);
                    } catch (Exception ex) {
                        System.out.println("Error: SendMessage - actionPerformed()");
                    }
                    setVisible(false);
                }else{
                    statusBBS.insert("Please check each in the form.\n\n",0);
                }

            }

            if(ae.getSource() == can) {
                setVisible(false);
            }


        }

        private class WL extends WindowAdapter {
            // If window closes, exit
            public void windowClosing(WindowEvent e) {
                setVisible(false);
            }
        }

    }




    /*
    *   Create a dialog to join a group and validate with the WebCOM Server
    */

    private class JoinGroup extends Dialog implements ActionListener {
        private Button ok,can;
        private TextField username;
        private TextField password;
        private TextField ipaddress;
        private TextField portMcast;
        private TextField ttlMcast;
        private TextField groupNameJoin;
        private TextField applicationJoin;


        private String strChat;
        private String strWB;
        private String chkStrChat;
        private String chkStrWB;

        private boolean chkAppTypeChat;
        private boolean chkAppTypeWB;




        JoinGroup(Frame frame){
            super(frame, "Joining a Group", true);
            setLayout(new FlowLayout());

            groupNameJoin = new TextField(15);
            applicationJoin = new TextField(15);
            ipaddress = new TextField(15);
            portMcast = new TextField(15);
            ttlMcast = new TextField(15);
            username = new TextField(15);
            password = new TextField(15);

            password.setEchoChar('*');


            add(new Label("Group Name:"));
            add(groupNameJoin);

            add(new Label("Chat/Whiteboard:"));
            add(applicationJoin);

            add(new Label("IP address :"));
            add(ipaddress);

            add(new Label("Port Number :"));
            add(portMcast);

            add(new Label("TTL Value :"));
            add(ttlMcast);

            add( new Label("User Name:"));
            add(username);

            add( new Label("Password :"));
            add(password);

            addOKCancelPanel();
            addWindowListener(new WL());
            createFrame();
            //pack();
            //setVisible(true);
        }

        void addOKCancelPanel() {
            Panel p = new Panel();
            p.setLayout(new FlowLayout());
            createButtons( p );
            add( p );
        }

        void createButtons(Panel p) {
            p.add(ok = new Button("Join"));
            ok.addActionListener(this);
            p.add(can = new Button("Cancel"));
            can.addActionListener(this);
        }

        void createFrame() {
            Dimension d = getToolkit().getScreenSize();
            setLocation(d.width/2,d.height/3);
            setResizable(false);
            setSize(160,480);
            show();
        }

        public void actionPerformed(ActionEvent ae){
            if(ae.getSource() == ok) {
                strChat = applicationJoin.getText();
                strWB = applicationJoin.getText();
                chkStrChat = "Chat";
                chkStrWB = "Whiteboard";

                chkAppTypeChat =  chkStrChat.equals(strChat);
                chkAppTypeWB =  chkStrWB.equals(strWB);

                System.out.println("Chat status : "+chkAppTypeChat);
                System.out.println("WB status : "+chkAppTypeWB);



                if (chkAppTypeChat == true){
                    statusBBS.insert("Loading a WebCOM Chat system.  Please WAIT...\n\n",0);
                    setVisible(false);
                    try {
                        WebCOMChat chatUserManual = new WebCOMChat(ipaddress.getText(),portMcast.getText(),ttlMcast.getText());
                    }catch (Exception ex){ex.toString();}
                } else if (chkAppTypeWB == true){
                    statusBBS.insert("Loading a WebCOM Whiteboard system.  Please WAIT...\n\n",0);
                    setVisible(false);
                    try{
                        WebCOMWhiteboard whiteboardUserManual = new
                            WebCOMWhiteboard(ipaddress.getText(),portMcast.getText(),ttlMcast.getText());

                    }catch (Exception ex){ex.toString();}
                }else{
                    statusBBS.insert("Please check each entry in the form.\n\n",0);
                }
            }

            if(ae.getSource() == can) {
                setVisible(false);
            }
        }


        private class WL extends WindowAdapter {
            // If window closes, exit
            public void windowClosing(WindowEvent e) {
                setVisible(false);
            }
        }
    }



    /*
    *   Create a dialog for creating a new group into the system
    *   and send to the server for storing into the database
    */

    private class CreateNewGroup extends Dialog implements ActionListener {
        private boolean idd = false;
        private Button okay,cancel;
        private TextField groupNameNew;
        private TextField dateStart;
        private TextField dateEnd;
        private TextField timeStart;
        private TextField timeEnd;
        private TextField applications;
        private TextField ipaddressCreateNew;
        private TextField portCreateNew;
        private TextField ttlCreateNew;
        private TextField moderatorName;
        private TextField moderatorPassword;




        CreateNewGroup(Frame frame){
            super(frame, "Create a New Group", true);
            setLayout(new FlowLayout());

            groupNameNew = new TextField(15);
            dateStart = new TextField(15);
            dateEnd = new TextField(15);
            timeStart = new TextField(15);
            timeEnd = new TextField(15);
            applications = new  TextField(15);

            ipaddressCreateNew = new TextField(15);
            portCreateNew = new TextField(15);
            ttlCreateNew = new TextField(15);
            moderatorName = new TextField(15);
            moderatorPassword = new TextField(15);

            add( new Label("Group Name :                    "));
            add(groupNameNew);

            add( new Label("Date Start :                        "));
            add(dateStart);
            add( new Label("Time Start :                       "));
            add(timeStart);
            add( new Label("Date Finish:                       "));
            add(dateEnd);
            add( new Label("Time Finish:                       "));
            add(timeEnd);


            add( new Label("Chat/Whiteboard:             "));
            add(applications);

            add(new Label("IP address :                    "));
            add(ipaddressCreateNew);

            add(new Label("Port Number :                   "));
            add(portCreateNew);

            add(new Label("TTL  Value :                     "));
            add(ttlCreateNew);

            add(new Label("Moderator User Name  :"));
            add(moderatorName);

            add(new Label("Moderator Password :     "));
            add(moderatorPassword);



            addOKCancelPanel();
            addWindowListener(new WL());
            createFrame();
            //pack();
            //setVisible(true);
        }

        void addOKCancelPanel() {
            Panel pp = new Panel();
            pp.setLayout(new FlowLayout());
            createButtons( pp );
            add( pp );
        }

        void createButtons(Panel pp) {
            pp.add(okay = new Button("Create"));
            okay.addActionListener(this);
            pp.add(cancel = new Button("Cancel"));
            cancel.addActionListener(this);
        }

        void createFrame() {
            Dimension dd = getToolkit().getScreenSize();
            setLocation(dd.width/2,dd.height/3);
            setResizable(false);
            setSize(310,410);
            show();
        }

        public void actionPerformed(ActionEvent ae){

            if(ae.getSource() == okay) {
                if(groupNameNew.getText()!=null &&
                    dateStart.getText()!=null &&
                    timeStart.getText()!=null &&
                    dateEnd.getText()!=null &&
                    timeEnd.getText()!=null &&
                    applications.getText()!=null &&
                    ipaddressCreateNew.getText()!=null &&
                    portCreateNew.getText()!=null &&
                    ttlCreateNew.getText()!=null &&
                    moderatorName.getText()!=null &&
                    moderatorPassword.getText()!=null )
                {
                    statusBBS.insert("Creating New Group. Please Wait...\n\n",0);
                    String info =   groupNameNew.getText()+";"+
                                    dateStart.getText()+";"+
                                    timeStart.getText()+";"+
                                    dateEnd.getText()+";"+
                                    timeEnd.getText()+";"+
                                    applications.getText()+";"+
                                    ipaddressCreateNew.getText()+";"+
                                    portCreateNew.getText()+";"+
                                    ttlCreateNew.getText()+";"+
                                    moderatorName.getText()+";"+
                                    moderatorPassword.getText()+";";
                    String message = "NEWGROUP000000000000;"+info;
                    try
                    {
                        byte[] data = message.getBytes("UTF8");
                        DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
                        socket.send(dp);
                    } catch (Exception ex) {
                        System.out.println("Error: SendMessage - actionPerformed()");
                    }
                    setVisible(false);
                }else{
                    statusBBS.insert("Please check each entry in the form.\n\n",0);
                }
            }
            if(ae.getSource() == cancel) {
                setVisible(false);
            }
        }
        private class WL extends WindowAdapter {
            // If window closes, exit
            public void windowClosing(WindowEvent e) {
                setVisible(false);
            }
        }

      }

    //close the current windows
    public void close() {
         JSObject win = (JSObject)JSObject.getWindow(this);
        win.eval("window.opener=self;self.close();");
    }

    public void run(){
        Thread thisThread = Thread.currentThread();
        while (receiver == thisThread) {
            try {
                thisThread.sleep(1);
            } catch (InterruptedException e){
            }
        }
    }

   public void stop(){
      receiver = null;
      try{
        if (receiver != null) receiver = null;
         if (socket != null)  socket.close();

      }catch(Exception ioe){
         System.out.println(ioe.toString());
      }
}
```

J.3 WebCOM Chat

```
/*
*   Md Asri Ngadi (c) 2004
*   WebCOM Client - Chat System
*   Create a user interface for WebCOM System allow user communicate using Chat System.
*
*   Most of the coding is from the author.  Some of them has obtained from various sources
*   on the Internet and modified according to the objective of the research.
*
*
*/

package com.sun.multicast.reliable.applications.BBS;

import javax.swing.*;
import java.io.*;
import java.net.*;
import java.awt.event.*;
import java.awt.*;

import java.lang.String;
import com.sun.multicast.util.*;
import com.sun.multicast.reliable.*;
import com.sun.multicast.reliable.transport.*;
import com.sun.multicast.reliable.transport.lrmp.*;
import com.sun.multicast.reliable.transport.lrmp.LRMPPacketSocket;


public class WebCOMChat extends JApplet implements
ActionListener,Runnable {


    private JTextField messageFieldChat;
    private TextArea chatAreaChat;

    private RMPacketSocket socketChat = null;
    private ReceiverThread receiverChat = null;
    private String transportName = "LRMP";
    private String prefixChat;
    private String dummyPortChat="";
    private String dummyIPChat="";
    private int dummyTTLChat;
    private int portChat;
    private InetAddress iaChat;
    private boolean done = false;
    private Frame boardChat;
    private Button exitChat;
    private Font ver = new Font("Verdana",1,10);
    private List statusMember = new List(10);

    /*
    *   Constructor for WebCOM Chat
    */

    public WebCOMChat(String ip4Chat, String port4Chat,String TTLChat)
    {
        dummyPortChat = port4Chat;
        dummyIPChat = ip4Chat;
        dummyTTLChat = Integer.parseInt(TTLChat);
        init();
    }

    public void init(){



        exitChat = new Button("Exit");   // The third button.
        exitChat.addActionListener(this);
        exitChat.setBackground(Color.lightGray);


        //create CHAT area for whiteboard
        Panel panelChat = new Panel();
        panelChat.setLayout(new BorderLayout());
        messageFieldChat = new JTextField();
        messageFieldChat.addActionListener(new SendMessageChat());
        chatAreaChat = new TextArea();
        chatAreaChat.setEditable(false);
        //panelChat.add("North", chatAreaChat);
        panelChat.add("West", new Label("Enter a message:"));
        panelChat.add("Center", messageFieldChat);
        panelChat.add("East", exitChat);

        statusMember = new List(10);
        statusMember.setBounds(5,5,50,50);
        statusMember.setFont(ver);

        //create a board for all components of whiteboard
        boardChat = new Frame();
        boardChat.setSize(600,400);
        boardChat.addWindowListener(new WL());
        boardChat.setResizable(false);
        boardChat.setTitle("WebCOMChat for " + UserNameSystem);

        //add components
        boardChat.add(chatAreaChat, "Center");
        boardChat.add(panelChat, "South");
        boardChat.add(statusMember, "East");
        boardChat.show();
        messageFieldChat.requestFocus();

        /*
        *   Joining a group
        */
        try{
            iaChat = InetAddress.getByName(dummyIPChat);
        }catch (UnknownHostException e){
            throw new IllegalArgumentException();
        }

        prefixChat = UserNameSystem + ">>";
        portChat = Integer.parseInt(dummyPortChat);

        //call socket creation
        try{
                createSocketChat();
        } catch (Exception ex){
                System.out.println("Error: Creating the socket.");
        }

        //Send to other participants - sign in
        sendSignInChat();
    }

    public void close() {

        if (boardChat != null) {
            boardChat.dispose();
        }

        sendSignoffChat();
    }
    //Inner class to tell us when the window is closing.
    private class WL extends WindowAdapter {
        // If window closes, exit
        public void windowClosing(WindowEvent e) {
            synchronized (WebCOMChat.this) {
                done = true;
                WebCOMChat.this.notifyAll();
            }
            close();
        }
    }

    /*
    *   Socket creation for WebCOM Chat
    */
    private void createSocketChat() throws IOException, RMException{

        TransportProfile tp = null;
        if (transportName.equals("LRMP")) {
            LRMPTransportProfile lrmptp = null;
            try {
                lrmptp = (LRMPTransportProfile) new LRMPTransportProfile(iaChat, portChat);
            } catch (InvalidMulticastAddressException e) {
                throw new ImpossibleException(e);
            }
            lrmptp.setTTL((byte) dummyTTLChat);
            lrmptp.setOrdered(true);
            tp = lrmptp;

        } else {
            throw new AssertFailedException();
        }

        try {
            socketChat = tp.createRMPacketSocket(TransportProfile.SEND_RECEIVE);
        } catch (InvalidTransportProfileException ie) {
            System.out.println("Error: InvalidTransportProfileException");
            throw new ImpossibleException(ie);
        } catch (UnsupportedException ue) {
            System.out.println("Error: UnsupportedException");
            throw new ImpossibleException(ue);
        } catch (IOException ioe){
            System.out.println("Error: IOException");
        }
        catch (RMException rme){
            System.out.println("Error: RMException");
        }

        receiverChat = new ReceiverThread();
        receiverChat.start();
   }

          /**
    * Inner class to send a message.
    */
   private class SendMessageChat implements ActionListener {
        /**
         * When a message is entered, send it.
         */
        public void actionPerformed(ActionEvent e) {
        System.out.println("Masuk SendMessageChat()");
            try {
                String messageSend = "WEBCOMCHAT0000000000;"+prefixChat + messageFieldChat.getText();
                String messageView = prefixChat + messageFieldChat.getText();
                messageFieldChat.setText("");
                chatAreaChat.append(messageView + "\n");
                System.out.println("Send Message : "+messageSend);
                byte data[] = messageSend.getBytes("UTF8");
                DatagramPacket dp = new DatagramPacket(data, data.length, iaChat, portChat);

                socketChat.send(dp);
            } catch (Exception ex) {
                System.out.println("Error SendMessageChat : "+ex.toString());
            }
        }
    }


      /**
     * Inner class to receive messages and display them
     */
    private class ReceiverThread extends Thread {
        private boolean done = false;
        private boolean statusChat = false;
        private boolean statusSignOff = false;
        private boolean statusSignIn = false;
        private String chkStatusChat           = "WEBCOMCHAT0000000000";
        private String chkSignOff              = "signing off";
        private String chkSignIn              = "has signing in";

        public void run() {
            while (!done) {

                try {
                    //receive message
                    DatagramPacket dp = socketChat.receive();
                    String message = new String(dp.getData(), "UTF8");

                    //put the message into string buffer
                    StringBuffer strBfr = new StringBuffer(message);
                    String chkStatus = strBfr.substring(0,20);
                    strBfr = strBfr.delete(0,21);

                    //check the WORD control
                    statusChat = chkStatusChat.equals(chkStatus);

                    //if message is for WebCOM Chat display it
                    if (statusChat == true){

                        chatAreaChat.append(strBfr + "\n");
                    }


                    //display status of other user
                    chkStatus = strBfr.substring(strBfr.length()-11,strBfr.length());
                    statusSignOff = chkSignOff.equals(chkStatus);
                    if (statusSignOff == true){
                        int j = strBfr.indexOf(">>");
                        String member = strBfr.substring(0,j);
                        statusMember.add(member+" is not active");
                    }

                    chkStatus = strBfr.substring(strBfr.length()-14,strBfr.length());
                    statusSignIn = chkSignIn.equals(chkStatus);
                    if (statusSignIn == true){
                        int j = strBfr.indexOf(">>");
                        String member = strBfr.substring(0,j);
                        statusMember.add(member + " is active");
                    }


                    chkStatus = strBfr.substring(strBfr.length()-11,strBfr.length());
                    statusSignOff = chkSignOff.equals(chkStatus);


                } catch (Exception ex) {}
            }
        }

        public void terminate() {
            done = true;
        }
    }


    /**
    *  Sends a signoff message to other users when window is closing
    */
    private void sendSignoffChat() {
        String message = "WEBCOMCHAT0000000000;"+prefixChat + " signing off";
        try {
            byte data[] = message.getBytes("UTF8");
            DatagramPacket dp = new DatagramPacket(data, data.length, iaChat, portChat);
            socketChat.send(dp);
        } catch (Exception ex) {}
    }

    /**
    *  Sends a signIn message to other users when window is closing
    */
    private void sendSignInChat() {
        String message = "WEBCOMCHAT0000000000;"+ prefixChat+ " has signing in";

        try {
            byte data[] = message.getBytes("UTF8");
            DatagramPacket dp = new DatagramPacket(data, data.length, iaChat, portChat);
            socketChat.send(dp);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    public void run() {
    }
    //exit when the "Exit" button pressed
    public void actionPerformed(ActionEvent e){
        if (e.getSource() == exitChat){
            close();
        }
    }

}
```

J.4 WebCOM Whiteboard

```

/*
*   Md Asri Ngadi (c) 2004
*   WebCOM Client - Whiteboard System
*   Create a user interface for WebCOM System allow user to login to the WebCOM System,
*   register a user into the system, create a new group for moderator (please request one
*   from administrator), request events happening in the system, join any group on the
*   list or a specified known group.
*
*   Most of the coding is from my self and some of them has obtained from various sources
*   on the Internet and modified according to the objective of the research.
*
*
*/

package com.sun.multicast.reliable.applications.BBS;

import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import java.lang.StringBuffer;

import javax.swing.*;
import java.io.*;
import java.net.*;
import java.net.URL;
import java.awt.*;
import java.awt.event.*;
import netscape.javascript.*;
import javax.swing.text.InternationalFormatter;
import java.lang.String;

import com.sun.multicast.util.*;
import com.sun.multicast.reliable.*;
import com.sun.multicast.reliable.transport.*;
import com.sun.multicast.reliable.transport.lrmp.*;
import com.sun.multicast.reliable.transport.lrmp.LRMPPacketSocket;

public class WebCOMWhiteboard extends Applet
    implements ActionListener, MouseListener, MouseMotionListener, KeyListener
{
    //declare class objects
    private CloseableFrame board;
    private Canvas canvas;
    private Point oldPoint, currentPoint;
    private TextField message;
    private String s;
    private TextField messageFieldWB;
    private TextArea chatAreaWB;

    private RMPacketSocket socketWB = null;
    private ReceiverThreadWB receiverWB = null;
    private Font ver = new Font("Verdana",1,12);
    private Font ver2 = new Font("Verdana",1,11);
    private String transportName = "LRMP";
    private String prefixWB;
    private String dummyPortWB="";
    private String dummyIPWB="";
    private String dummyTTLWB="";
    private int portWB;
    private InetAddress iaWB;
    private String UserNameSystemWB;
    private boolean done = false;
    private int colorSelect;
    private Button exit, clear, fill,foreGround;

    private boolean dragging;     // This is set to true when the user is drawing.

    private int figure;    // What type of figure is being drawn.  This is
                                //    specified by the figureChoice menu.


    private final static int
        BLACK = 0,
        RED = 1,            // Some constants to make
        GREEN = 2,          // the code more readable.
        BLUE = 3,           // These numbers code for
        CYAN = 4,           // the differnet drawing colors.
        MAGENTA = 5,
        YELLOW = 6,
        WHITE = 7;

    Choice colorChoice;  // A Choice object, containing the possible drawing
                        // colors, which must be created by the applet.

    private final static int
        CURVE = 0,
        LINE = 1,
        RECT = 2,              // Some constants that code
        OVAL = 3,              // for the different types of
        ROUNDRECT = 4,         // figure the program can draw.
        FILLED_RECT = 5,
        FILLED_OVAL = 6,
        FILLED_ROUNDRECT = 7;
```

```java

    Choice figureChoice;  // A Choice object containg the possible figures.


    public WebCOMWhiteboard(String ip4WB, String port4WB, String ttl4WB, String name4WB)
    {
        dummyPortWB = port4WB;
        dummyIPWB = ip4WB;
        dummyTTLWB = ttl4WB;
        UserNameSystemWB = name4WB;
        init();
    }

    //the starting point of the applet
    public void init() {

        try{
            iaWB = InetAddress.getByName(dummyIPWB);
        }catch (UnknownHostException e){
            throw new IllegalArgumentException();
        }

        prefixWB = UserNameSystemWB + ">>";
        portWB = Integer.parseInt(dummyPortWB);

        try{
            createSocketWB();
        } catch (Exception ex){
            System.out.println("Error: Creating the socket.");

        }

        //initialize the "old" point
        oldPoint = new Point(0,0);

        //add a button to the panel
        //clear = new Button("Clear");
        //clear.addActionListener(this);
        Panel control = new Panel();
        control.setBackground(Color.lightGray);

        Panel buttonBar = new Panel();       // A panel to hold the buttons.
        buttonBar.setBackground(Color.lightGray);
        control.add(buttonBar);

        Panel choiceBar = new Panel();       // A panel to hole the pop-up menus
        choiceBar.setBackground(Color.lightGray);
        control.add(choiceBar);


        // The first button.
        fill = new Button("Set Background");
        fill.addActionListener(this);
        fill.setBackground(Color.lightGray);
        buttonBar.add(fill);

        // The second button.
        foreGround = new Button("Set Foreground");
        foreGround.addActionListener(this);
        foreGround.setBackground(Color.lightGray);
        buttonBar.add(foreGround);

        // The third button.
        clear = new Button("Clear");   // The second button.
        clear.addActionListener(this);
        clear.setBackground(Color.lightGray);
        buttonBar.add(clear);

        // The fourth button.
        exit = new Button("Exit");   // The third button.
        exit.addActionListener(this);
        exit.setBackground(Color.lightGray);
        buttonBar.add(exit);


        Choice choice = new Choice();  // The pop-up menu of colors.
        choice.addItem("Black");
        choice.addItem("Red");
        choice.addItem("Green");
        choice.addItem("Blue");
        choice.addItem("Cyan");
        choice.addItem("Magenta");
        choice.addItem("Yellow");
        choice.addItem("White");
        choice.setBackground(Color.white);
        choiceBar.add(choice);


        Choice choice2 = new Choice();  // The pop-up menu of shapes.
        choice2.addItem("Curve");
        choice2.addItem("Straight Line");
        choice2.addItem("Rectangle");
        choice2.addItem("Oval");
        choice2.addItem("RoundRect");
        choice2.addItem("Filled Rectangle");
        choice2.addItem("Filled Oval");
```

```
            choice2.addItem("Filled RoundRect");
            choice2.setBackground(Color.white);
            choiceBar.add(choice2);

            colorChoice = choice;    // Canvas needs access to the pop-up menus,
            figureChoice = choice2;  //   so it can check it to find out what
                                     //   color and shape to use.

            //add a canvas for painting
            canvas = new Canvas();
            canvas.setBackground(Color.white);

            //add listeners
            canvas.addMouseListener(this);
            canvas.addMouseMotionListener(this);
            canvas.addKeyListener(this);

            //add a message center to tell the user the cursor's coordinates
            message = new TextField("Draw away! Click clear to start over.");
            message.setBackground(Color.gray);

            //create CHAT area for whiteboard
            Panel p = new Panel();
            p.setLayout(new BorderLayout());
            messageFieldWB = new TextField();
            messageFieldWB.addActionListener(new SendMessageWB());
            chatAreaWB = new TextArea();
            chatAreaWB.setEditable(false);
            p.add("North", chatAreaWB);
            p.add("West", new Label("Enter a message:"));
            p.add("Center", messageFieldWB);
            p.add("South",message);
            messageFieldWB.requestFocus();

            //create the window for the whiteboard
            board = new CloseableFrame("WebCOM Whiteboard: "+ UserNameSystemWB);
            board.setLayout(new BorderLayout());
            board.setSize(600,600);
            board.setResizable(false);
            board.setVisible(true);

            //add components
            board.add(control, "North");
            board.add(canvas, "Center");
            board.add(p, "South");
            board.validate();


        }

        /*
        *   Create the socket
        */

        private void createSocketWB() throws IOException, RMException{
            TransportProfile tp = null;
            if (transportName.equals("LRMP")) {
                LRMPTransportProfile lrmptp = null;
                try {
                    lrmptp = (LRMPTransportProfile) new LRMPTransportProfile(iaWB, portWB);
                } catch (InvalidMulticastAddressException e) {
                    throw new ImpossibleException(e);
                }
                lrmptp.setTTL((byte)Integer.parseInt(dummyTTLWB) );
                lrmptp.setOrdered(false);
                //lrmptp.setMaxDataRate(100000);
                tp = lrmptp;

            } else {
                throw new AssertFailedException();
            }

            try {
                socketWB = tp.createRMPacketSocket(TransportProfile.SEND_RECEIVE);
            } catch (InvalidTransportProfileException ie) {
                System.out.println("Error: InvalidTransportProfileException");
                throw new ImpossibleException(ie);
            } catch (UnsupportedException ue) {
                System.out.println("Error: UnsupportedException");
                throw new ImpossibleException(ue);
            } catch (IOException ioe){
                System.out.println("Error: IOException");
            }
            catch (RMException rme){
                System.out.println("Error: RMException");
            }

            receiverWB = new ReceiverThreadWB();
            receiverWB.start();
        }


         /**
         * Inner class to send a message.
         */
        private class SendMessageWB implements ActionListener {
```

```
        /**
         * When a message is entered, send it.
         */
        public void actionPerformed(ActionEvent e) {
        System.out.println("Masuk SendMessage()");
            try {
                String messageSend = "CHATWB00000000000000;"+prefixWB + messageFieldWB.getText();
                String messageView = prefixWB + messageFieldWB.getText();
                messageFieldWB.setText("");
                chatAreaWB.append(messageView + "\n");

                byte data[] = messageSend.getBytes("UTF8");
                DatagramPacket dp = new DatagramPacket(data, data.length, iaWB, portWB);

                socketWB.send(dp);
            } catch (Exception ex) {
                System.out.println(ex.toString());
            }
        }
    }

   /**
    * Inner class to receive data, filter and display them on canvas
    */
    private class ReceiverThreadWB extends Thread {
        private boolean done = false;
        private boolean statusChatWB = false;
        private boolean statusObjectDraw = false;
        private boolean statusKeyTyped = false;

        private String chkStatusChatWB          = "CHATWB00000000000000";
        private String chkStatusObjectDraw      = "OBJECTDRAW0000000000";
        private String chkStatusKeyTyped        = "DATAKEYTYPED00000000";


        public void run() {
        System.out.println("Masuk ReceiverThread");
            while (!done) {

                try {

                    DatagramPacket dp = socketWB.receive();
                    String message = new String(dp.getData(), "UTF8");

                    //System.out.println("Receive data from client : "+message);

                    StringBuffer strBfr = new StringBuffer(message);
                    String chkStatus = strBfr.substring(0,20);
                    strBfr = strBfr.delete(0,21);

                    statusChatWB = chkStatusChatWB.equals(chkStatus);
                    statusObjectDraw = chkStatusObjectDraw.equals(chkStatus);
                    statusKeyTyped = chkStatusKeyTyped.equals(chkStatus);


                    if (statusChatWB == true){
                        chatAreaWB.append(strBfr + "\n");
                    }
                    if (statusObjectDraw == true){

                        int z = strBfr.indexOf(";");
                        String colorBuffer = strBfr.substring(0,z);
                        int colorR = Integer.parseInt(colorBuffer);
                        strBfr = strBfr.delete(0,z+1);

                        z = strBfr.indexOf(";");
                        String kindBuffer = strBfr.substring(0,z);
                        int kindR = Integer.parseInt(kindBuffer);
                        strBfr = strBfr.delete(0,z+1);

                        z = strBfr.indexOf(";");
                        String currentPointXreceive = strBfr.substring(0,z);
                        int currentPointXR = Integer.parseInt(currentPointXreceive);
                        strBfr = strBfr.delete(0,z+1);

                        z = strBfr.indexOf(";");
                        String currentPointYreceive = strBfr.substring(0,z);
                        int currentPointYR = Integer.parseInt(currentPointYreceive);
                        strBfr = strBfr.delete(0,z+1);

                        z = strBfr.indexOf(";");
                        String oldPointXreceive = strBfr.substring(0,z);
                        int oldPointXR = Integer.parseInt(oldPointXreceive);
                        strBfr = strBfr.delete(0,z+1);

                        z = strBfr.indexOf(";");
                        String oldPointYreceive = strBfr.substring(0,z);
                        int oldPointYR = Integer.parseInt(oldPointYreceive);
                        strBfr = strBfr.delete(0,z+1);

                        z = strBfr.indexOf(";");
                        String outlineOnlyReceive = strBfr.substring(0,z);
                        strBfr = strBfr.delete(0,z+1);

                        boolean outlineOnlyR=false;

                if (outlineOnlyReceive.equals("false")) {
                    outlineOnlyR=false;
                }
                if (outlineOnlyReceive.equals("true")) {
                    outlineOnlyR=true;
                }

            //if (kindR == LINE){
            if (kindR == 1){
                canvas.getGraphics().drawLine(currentPointXR, currentPointYR, oldPointXR, oldPointYR);

            }
            else {
                int x, y, w, h;  // Top-left corner, width, and height.
                if (oldPointXR >= currentPointXR) {  // x1 is left edge
                    x = currentPointXR;
                    w = oldPointXR - currentPointXR;

                }
                else {          // x2 is left edge
                    x = oldPointXR;
                    w = currentPointXR - oldPointXR;
                }

                if (oldPointYR >= currentPointYR) {  // y1 is top edge
                    y = currentPointYR;
                    h = oldPointYR - currentPointYR;
                }
                else {          // y2 is top edge.
                    y = oldPointYR;
                    h = currentPointYR - oldPointYR;
                }
                switch (kindR) {  // Draw the appropriate figure.
                    //case RECT:
                    case 2:
                    canvas.getGraphics().drawRect(x, y, w, h);

                    break;
                    //case OVAL:
                    case 3:
                    canvas.getGraphics().drawOval(x, y, w, h);
                    break;
                    //case ROUNDRECT:
                    case 4:
                    canvas.getGraphics().drawRoundRect(x, y, w, h, 20, 20);
                    break;
                    //case FILLED_RECT:
                    case 5:
                    if (outlineOnlyR)
                    {
                        canvas.getGraphics().drawRect(x, y, w, h);
                    } else {
                        canvas.getGraphics().fillRect(x, y, w, h);
                    }
                    break;
                    //case FILLED_OVAL:
                    case 6:
                    if (outlineOnlyR){
                        canvas.getGraphics().drawOval(x, y, w, h);
                    }
                    else {
                        canvas.getGraphics().fillOval(x, y, w, h);
                    }
                    break;
                    //case FILLED_ROUNDRECT:
                    case 7:
                    if (outlineOnlyR){
                        canvas.getGraphics().drawRoundRect(x, y, w, h, 20, 20);
                    }
                    else {
                        canvas.getGraphics().fillRoundRect(x, y, w, h, 20, 20);
                    }
                    break;
                }
            }

        }

        if (statusKeyTyped == true){
            int z = strBfr.indexOf(";");
            String Sreceive = strBfr.substring(0,z);
            String Str = Sreceive;
            strBfr = strBfr.delete(0,z+1);

            z = strBfr.indexOf(";");
            String oldPointXreceive = strBfr.substring(0,z);
            int oldPointX = Integer.parseInt(oldPointXreceive);
            strBfr = strBfr.delete(0,z+1);

            z = strBfr.indexOf(";");
            String oldPointYreceive = strBfr.substring(0,z);
            int oldPointY = Integer.parseInt(oldPointYreceive);
            strBfr = strBfr.delete(0,z+1);

            canvas.getGraphics().drawString(Str, oldPointX, oldPointY);
        }
```

```


                } catch (Exception ex) {}

            }
        }

        public void terminate() {
            done = true;
        }
    }


        public void putFigure(int kind,
                                  int x1, int y1, int x2, int y2, boolean outlineOnly) {
                // Draws a figure with corners at (x1,y1) and (x2,y2).  The
                // parameter "kind" codes for the type of figure to draw.  If the
                // figure is LINE, a line is drawn between the points.  For the
                // other shapes, we need the top-left corner, the width, and the
                // height of the shape.  We have to figure out whether x1 or x2
                // is the left edge of the shape and compute the width accordingly.
                // Similarly for y1 and y2.  If outlineOnly is true, then filled shapes
                // are drawn in outline only.

            if (kind == LINE){
                canvas.getGraphics().drawLine(x1, y1, x2, y2);

            }
            else {
                int x, y, w, h;  // Top-left corner, width, and height.
                if (x2 >= x1) {  // x1 is left edge
                    x = x1;
                    w = x2 - x1;
                }
                else {          // x2 is left edge
                    x = x2;
                    w = x1 - x2;
                }
                if (y2 >= y1) {  // y1 is top edge
                    y = y1;
                    h = y2 - y1;
                }
                else {          // y2 is top edge.
                    y = y2;
                    h = y1 - y2;
                }
                switch (kind) {  // Draw the appropriate figure.
                    case RECT:
                    canvas.getGraphics().drawRect(x, y, w, h);

                    break;
                    case OVAL:
                    canvas.getGraphics().drawOval(x, y, w, h);

                    break;
                    case ROUNDRECT:
                    canvas.getGraphics().drawRoundRect(x, y, w, h, 20, 20);

                    break;
                    case FILLED_RECT:
                    if (outlineOnly)
                    {
                        canvas.getGraphics().drawRect(x, y, w, h);

                    } else {
                        canvas.getGraphics().fillRect(x, y, w, h);


                    }
                    break;
                    case FILLED_OVAL:
                    if (outlineOnly){
                        canvas.getGraphics().drawOval(x, y, w, h);

                    }
                    else {
                        canvas.getGraphics().fillOval(x, y, w, h);
                    }
                    break;
                    case FILLED_ROUNDRECT:
                    if (outlineOnly){
                        canvas.getGraphics().drawRoundRect(x, y, w, h, 20, 20);

                    }
                    else {
                        canvas.getGraphics().fillRoundRect(x, y, w, h, 20, 20);

                    }
                    break;
                }
            }
        }


    public Color getCurrentColor() {
    // Check the colorChoice menu to find the currently
```

```java
    // selected color, and return the appropriate color
    // object.
        int currentColor = colorChoice.getSelectedIndex();
        switch (currentColor) {
            case BLACK:
                return Color.black;
            case RED:
                return Color.red;
            case GREEN:
                return Color.green;
            case BLUE:
                return Color.blue;
            case CYAN:
                return Color.cyan;
            case MAGENTA:
                return Color.magenta;
            case YELLOW:
                return Color.yellow;
            default:
                return Color.white;
        }
    }

    /*
    *   Send objects with their informations to other users
    */
    public void sendObject(int colorSend, int kindSend, int xSend,
        int ySend, int wSend, int hSend, boolean outlineOnlySend)
    {
        try {
            String messageSend = "OBJECTDRAW0000000000;"+colorSend+";"
                                 +kindSend+";"+xSend+";"+ySend+";"+wSend+";"+
                                 hSend+";"+outlineOnlySend+";";
            byte data[] = messageSend.getBytes("UTF8");

            DatagramPacket dp = new DatagramPacket(data, data.length, iaWB, portWB);
            socketWB.send(dp);
            } catch (Exception ex) {
                System.out.println(ex.toString());
            }
        }

    //actionPerformed listens to the button click
    public void actionPerformed(ActionEvent e){
        //if (e.getSource() instanceof Button){}
        if (e.getSource() == clear){
            canvas.repaint();
            canvas.getGraphics().setColor(Color.white);
            canvas.getGraphics().fillRect(0,0,600,600);
            message.setText("Cleared!");
        }
        //Change background
        if (e.getSource() == fill){
                    // Set background color, then clear.
            canvas.setBackground(getCurrentColor());
                //clearOSC();
            //repaint();
            message.setText("Change background!");
        }
        //Change foreground. Other objects will be cleared
        if (e.getSource() == foreGround){
            // Set background color, then clear.
            canvas.setForeground(getCurrentColor());
            //canvas.getGraphics().setColor(getCurrentColor());
            message.setText("Change foreground.  Other objects will be cleared!");
            //repaint();
        }
        //exit from the system
        if (e.getSource() == exit){
            close();
        }
    }


    //when the mouse is pressed, a point is formed
    public void mousePressed(MouseEvent e){
            s="";
            if (dragging == true)  // Ignore mouse presses that occur
                return;            //    when user is already drawing a curve.
                                    //    (This can happen if the user presses
                                    //    two mouse buttons at the same time.)

            oldPoint = currentPoint = e.getPoint();  // Save mouse coordinates.

            figure = figureChoice.getSelectedIndex(); //Type of figure begin drawn.

            colorSelect = colorChoice.getSelectedIndex();
            canvas.getGraphics().setColor(getCurrentColor());

            if (figure != CURVE) {
                //canvas.getGraphics().setXORMode(getBackground());
                putFigure(figure, currentPoint.x, currentPoint.y, currentPoint.x, currentPoint.y, true);

            }

            dragging = true;  // Start drawing.
    }

    public void mouseReleased(MouseEvent e){
            if (dragging == false)
                return;  // Nothing to do because the user isn't drawing.
            dragging = false;
            if (figure != CURVE) {
                    // Draw the final shape on the canvas
                    putFigure(figure, currentPoint.x, currentPoint.y, oldPoint.x, oldPoint.y, true);
                    sendObject(colorSelect, figure, currentPoint.x, currentPoint.y,oldPoint.x, oldPoint.y, true);

                if (currentPoint.x != oldPoint.x || currentPoint.y != oldPoint.y) {
                    putFigure(figure, currentPoint.x, currentPoint.y , oldPoint.x, oldPoint.y, false);
                    sendObject(colorSelect, figure, currentPoint.x, currentPoint.y, oldPoint.x, oldPoint.y, false);
                }
            }
    }


    //mouseDragged will do the line drawing
    public void mouseDragged(MouseEvent e){
            if (dragging == false)
                return;  // Nothing to do because the user isn't drawing.
            currentPoint = e.getPoint();
            int x = currentPoint.x;
            int y = currentPoint.y;
            if (figure == CURVE) {
                // Draw the line on the canvas.
                putFigure(LINE, oldPoint.x, oldPoint.y, x, y, false);
                sendObject(colorSelect, 1, currentPoint.x, currentPoint.y, oldPoint.x, oldPoint.y, false);
                oldPoint.x = x;
                oldPoint.y = y;
            }
            else {
            }

         message.setText("Mouse Dragged: (" + currentPoint.x + ", " + currentPoint.y + ")");
    }



    //abstract methods promised by the MouseListener and MouseMotionListener interfaces
    public void mouseEntered(MouseEvent e){}
    public void mouseExited(MouseEvent e){}
    public void mouseClicked(MouseEvent e){}
    public void mouseMoved(MouseEvent e){}

    //Send a character to users when released a key
    public void keyReleased(KeyEvent e){

        try {
            String datakeyTyped =   "DATAKEYTYPED00000000;" + s + ";" +
                                   oldPoint.x + ";" + oldPoint.y + ";";
            byte data[] = datakeyTyped.getBytes("UTF8");
            DatagramPacket dp = new DatagramPacket(data, data.length, iaWB, portWB);
            socketWB.send(dp);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    //keyTyped will print the character, starting at the last place the mouse was pressed
    public void keyTyped(KeyEvent e){
        s += e.getKeyChar();
        canvas.getGraphics().drawString(s, oldPoint.x, oldPoint.y);
        message.setText( "Key typed." );

    }

    public void keyPressed(KeyEvent e){
    }

    /**
     * Sends a final signoff message when window is closing
     */
    private void sendSignoff() {
        String message = prefixWB + " signing off";
        try {
            byte data[] = message.getBytes("UTF8");
            DatagramPacket dp = new DatagramPacket(data, data.length, iaWB, portWB);
            socketWB.send(dp);
        } catch (Exception ex) {}
    }

    private void sendSignIn() {
        String message = UserNameSystemWB + " has signing In";
        try {
            byte data[] = message.getBytes("UTF8");
            DatagramPacket dp = new DatagramPacket(data, data.length, iaWB, portWB);

            socketWB.send(dp);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }

    /**
     * Closes a WebCOMWB object.
     */
    public void close() {
        if (board != null) {
            board.dispose();
        }

        if (receiverWB != null) {
            receiverWB.terminate();
        }
        if (socketWB != null) {
            sendSignoff();
            socketWB.close();
        }
    }

    public void run()
    {
        try {
            synchronized (this) {
                while (!done) {
                    wait();
                }
            }
        } catch (InterruptedException e) {}
        finally {
            close();
        }
    }

    public void start(){
        try {
            synchronized (this) {
                while (!done) {
                    wait();
                }
            }
        } catch (InterruptedException e) {}
        finally {
            close();
        }
    }

    public void stop(){
        close();
    }
}

class CloseableFrame extends Frame
{
    //the constructor sets up the background event processing
    public CloseableFrame(String title){
        super(title);
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);

    }
    //this method performs the event
    public void processWindowEvent(WindowEvent event){
        super.processWindowEvent(event);
        if(event.getID() == WindowEvent.WINDOW_CLOSING){
            dispose();
        }
    }
}
```